ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES

"DESARROLLO DE UN SISTEMA DE COMUNICACIÓN CON NFC PARA EL ACCESO A INFORMACIÓN ACADÉMICA DE LOS ESTUDIANTES DE LA FIE-ESPOCH"

Trabajo de titulación presentado para optar el grado académico de:

INGENIERO EN ELECTRÓNICA, TELECOMUNICACIONES Y REDES

AUTORES: DANIELA KARINA LARA TUZ

ROBERTO JAVIER VALLEJO MOLINA

TUTOR: ING. SANTIAGO CISNEROS

Riobamba – Ecuador

2016

ESCUELA SUPERIOR POLITÉCNICA DE CHIMBORAZO

FACULTAD DE INFORMÁTICA Y ELECTRÓNICA

ESCUELA DE INGENIERÍA ELECTRÓNICA EN TELECOMUNICACIONES Y REDES

El tribunal del Trabajo de Titulación certifica que: El trabajo de investigación: **DESARROLLO DE UN SISTEMA DE COMUNICACIÓN CON NFC PARA EL ACCESO A INFORMACIÓN ACADÉMICA DE LOS ESTUDIANTES DE LA FIE-ESPOCH**, de responsabilidad de los señores Daniela Karina Lara Tuz y Roberto Javier Vallejo Molina, ha sido minuciosamente revisado por los Miembros del Tribunal, quedando autorizada su presentación.

NOMBRE	**FIRMA**	**FECHA**
Dr. Washington Luna Encalada **DECANO DE LA FACULTAD DE INFORMÁTICA Y ELECTRÓNICA**	_____	_____
Ing. Franklin Moreno **DIRECTOR DE ESCUELA DE ELECTRÓNICA, TELECOMUNICACIONES Y REDES**	_____	_____
Ing. Santiago Cisneros **DIRECTOR DE TRABAJO DE TITULACIÓN**	_____	_____
Ing. Jorge Yuquilema **MIEMBRO DEL TRIBUNAL**	_____	_____

Nosotros, Daniela Karina Lara Tuz; Roberto Javier Vallejo Molina, somos responsables de las ideas, doctrinas y resultados expuestos en este Trabajo de Titulación y el patrimonio intelectual de la Tesis de Grado pertenece a la Escuela Superior Politécnica de Chimborazo.

_____

Daniela Karina Lara Tuz

_____

Roberto Javier Vallejo Molina

DEDICATORIA

Dedico este trabajo a mi Dios, quien ha sabido guiarme en cada paso que doy, por enseñarme el valor de la vida y regalarme fuerzas para superar todos los obstáculos que se me han presentado sin desfallecer, por enseñarme a ser una mujer esforzada, valiente, por su provisión, su amor y por cumplir las promesas que un día hizo en mí, por hacerme entender que sin el nada soy.

A mis amados padres Washington y Gladys por su amor, apoyo, guía constante, porque gracias a su esfuerzo diario he podido cumplir mis sueños y ahora soy lo que soy, un Dios les pague por todo, porque muchas veces se han sacrificado ustedes para que no me falte a mí, porque sé que siempre estarán a mi lado festejando mis triunfos, ayudándome en mis más duras luchas y porque a pesar de las adversidades son la primera bendición que tiene mi vida.

A mi madrecita preciosa por su ayuda, afecto, consejos y sobre todo comprensión, por ser mi compañera de batalla, mi mejor amiga, mi inspiración, el primer motivo para salir adelante, mi ejemplo y mi guerrera incansable, por luchar conmigo de la mano día a día, por tener las palabras precisas en cada circunstancia de mi vida, mil gracias por siempre creer en mí aun cuando nadie lo hizo, porque nunca te importó nada de lo que nadie diga siempre te la has jugado todo por mí y por estar a mi lado amándome sin condiciones a pesar de mis errores, por eso y más te amo mucho mi preciosa.

A ti mi amado esposo Javy que aparte de ser mi compañero de tesis, has sido el pintor de mis sueños, porque contigo he vivido de la mano esta dura travesía, por tu paciencia, amor, dedicación, perseverancia, por enseñarme que el amor es sufrido, benigno, que no tiene envidia, no busca lo suyo, sino el bienestar de la persona amada, que todo lo puede, todo lo cree, todo lo espera, todo lo soporta, por estar a mi lado compartiendo los buenos y sobretodo malos momentos, por siempre tener esas palabras que dan felicidad a mi vida, agradezco a Dios por el maravilloso hombre que ha puesto a mi lado, por ese futuro tan anhelado que construiremos juntos y por eso siempre bendigo tu vida mi amor.

A ti mi amado angelito/a, mi bebito hermoso que te formas dentro de mí, porque no sabes la dicha que sentimos, cuánto te esperamos, anhelamos y deseamos que ya estés nosotros tus papitos que te amaremos siempre, porque desde ya eres nuestra fuerza para vencer las adversidades con la ayuda de nuestro Señor, bendigo tu vida mi chiquito porque sé que eres un hijo con promesa y que llegarás mucho más lejos que nosotros.

Y a mis hermanos Mauricio y Ricardo por los consejos, penas, alegrías vividas, porque siempre supe que me cuidaban y han estado allí, al igual que yo siempre estaré aquí para ustedes.

Dany

DEDICATORIA

Dedico este trabajo a mi Dios, quien supo guardarme a lo largo del camino, darme la fuerza para seguir adelante y no desmayar en las dificultades, por forjar mi caracter enseñándome a encarar las adversidades que se han puesto delante de mí, porque tú eres mi escudo, mi roca, mi fortaleza, por amor de tu nombre me conducirás y guiarás.

A mis 3 padres Greta, Vero y Hugo por su amor, guía y comprensión, porque gracias a ustedes soy el hombre que soy ahora, por sus enseñanzas y consejos que han sido la base de mis decisiones. A ustedes mis amados padres que gracias a su sudor he podido cumplir mis sueños, por mancharse sus manos para que las mías puedan estar limpias, mil gracias porque sé que estarán observando mis pasos por lejos que me vaya, porque contaré con su apoyo haga lo que haga y esté donde esté, porque me han demostrado que la vida es un reto, un desafío y un regalo.

A ti mi bendición, mi esposa, gracias por llegar a mi vida cuando más lo necesitaba, por enseñarme que no hay obstáculo que no podamos superar, por ser el motor que impulsa mi caminar, mi bastón en los momentos difíciles, mi ayuda idónea la mujer que Dios me permitió entregarle todo mi amor, por los momentos buenos así como los malos, teniendo la seguridad que nuestro amor lo puede todo.

A mi hijo/a, porque desde ya te amamos y esperamos ansiosos tu llegada, recuerda que te amo mucho, sigue adelante y nunca te dejes vencer aunque te digan que no se puede lograr las metas, tu puedes hacerlos con constancia, rectitud y con la ayuda de Dios.

Y a mis hermanos Estefy, Santy y Jeremy por ser ese toque de alegria y locura que necesita mi vida, por las peleas, lágrimas y sonrisas que hemos compartido, que a pesar de las edades no han sido barrera para demostrarnos lo mucho que nos amamos, recuerden que mis logors no son más que la base para que ustedes escalen mucho más alto.

Javy

AGRADECIMIENTO

En primer lugar a nuestro Padre Celestial, nuestra Roca y Fortaleza, por su amor excepcional, por regalarnos el don de la vida, los sueños, por caminar a nuestro lado siempre, por sus promesas, por cada bendición que nos regala día a día y permitirnos cumplir nuestra meta tan anhelada.

A nuestros padres Washington, Gladys, Hugo y Verónica que han sido el motor que nos impulsa a salir adelante, por su ejemplo, dedicación, por creer en nosotros y por su apoyo incondicional en cada etapa de nuestra vida.

A nuestra familia y amigos que han formado una parte muy importante de nuestras vidas, gracias por su amistad, compañía, esperanza, por las palabras de ánimo en todo momento y por esas manos extendidas desinteresadamente.

Un agradecimiento muy especial a nuestro Director el Ing. Santiago Cisneros por su ayuda, apoyo incondicional, su guía, paciencia y motivación, ya que su enseñanza académica y moral han sido de gran beneficio, a los Ingenieros Jorge Yuquilema, José Guerra por la supervisión y colaboración que nos han brindado.

Dany y Javier

TABLA DE CONTENIDO

ÍNDICE DE TABLAS

ÍNDICE DE FIGURAS

ÍNDICE DE GRÁFICOS

ÍNDICE DE ABREVIATURAS

ASK	Amplitude-Shift Keying (Modulación por desplazamiento de Amplitud)
BNF	Backus-Naur form (Metalenguaje de gramáticas libres o lenguajes formales)
CF	Chunk Flag (Bandera Fragmentada)
DSSS	Direct Sequence Spread Spectrum (Espectro Ensanchado por Secuencia Directa)
DC	Direct Current (Corriente Directa)
EAS	Electronic Article Surveillance (Vigilancia Electrónica de Artículos)
ECMA	European Computer Manufacturers Association (Asociación Europea de Fabricantes de Ordenadores)
EEPROM	Electrically Erasable Programmable Read-Only Memory (Memoria de sólo lectura Programable y borrable Eléctricamente)
E/S	Input/Output (Entrada/Salida)
ESPOCH	Escuela Superior Politécnica de Chimborazo
ETSI	European Telecommunications Standards Institute (Instituto Europeo de Normas de Telecomunicaciones)
FIE	Facultad de Informática y Electrónica
GFSK	Gaussian Frequency Shift Keying (Modulación por Desplazamiento de Frecuencia Gaussiana)
GPL	General Public License (Licencia Pública General)
HCI	Host Controller Interface (Interfaz de control de Host)
HTML	HyperText Markup Language (Lenguaje de Marcas de Hipertexto)
IBM	International Business Machines (Máquina de Negocios Internacionales)
ICSP	In Circuit Serial Programming (Programación en circuito serie)
ID	Identifier (Identificador)
IDE	Integrated Development Environment (Entorno de Desarrollo Integrado)
IDT	Integrated Device Technology (Tecnología de Dispositivo Integrada)
IEC	International Electrotechnical Commission (Comisión Electrotécnica Internacional)

IEEE	Institute of Electrical and Electronics Engineers (Instituto de Ingeniería Eléctrica y Electrónica)
IFF	Identification Friend or Foe (Identificación amigo o enemigo)
IL	ID_LENGTH (Longitud de Identificador)
IP	Internet Protocol (Protocolo de Internet)
ISM	Industrial, Scientific and Medical
ISO	International Organization for Standardization (Organización Internacional para la Estandarización)
IVREA	Interaction Design Institute (Instituto de Diseño Interactivo)
JDK	Java Development Kit (Kit de Desarrollo de Java)
KBPS	Kilobits por segundo
LAN	Local Área Network (Red de Área Local)
LLCP	Logical Link Control Protocol (Protocolo de Control de Enlace Lógico)
MB	Message Begin (Mensaje de Inicio)
MBPS	Megabits por segundo
ME	Message End (Mensaje Final)
MIME	Multipurpose Internet Mail Extensions (Extensiones Multipropósito de Correos de Internet)
MISO	Master In Slave Out (Línea de esclavo para el envío de datos al maestro)
MOSI	Master Out Slave In (línea principal para el envío de datos a los periféricos)
NDEF	NFC Data Exchange Format (Formato de Intercambio de Datos NFC)
NFC	Near Field Communication (Comunicación de campo cercano)
NFCIP-1	Near Field Communication Interface and Protocol-1 (Interfaz de comunicación de campo cercano y el Protocolo 1)
NFCIP-2	Near Field Communication Interface and Protocol-2 (Interfaz de comunicación de campo cercano y el Protocolo 2)
OFDM	Orthogonal Frequency Division Multiplexing (Multiplexación por División de Frecuencias Ortogonales)
OSI	Open System Interconnection (Interconexión de sistemas abiertos)
PC	Personal Compute (Computador Personal)
PCD	Proximity Coupling Device (Dispositivo de acoplamiento de proximidad)

PHP	Hypertext Preprocessor (Preprocesador de Hipertexto)
PICC	Proximity Integrated Circuit Card (Tarjeta de circuito integrado de proximidad)
PWM	Pulse Width Modulation (Modulación por Ancho de pulsos)
P2P	Peer to Peer (Modo punto a punto)
QPSK	Quadrature Phase Shift Keying (Modulación por Desplazamiento de Fase Cuaternaria)
RF	Radio Frequency (Radiofrecuencia)
RFC	Request for Comments (Solicitud de Comentarios)
RFID	Radio Frequency Identification (Identificación por Radio Frecuencia)
RTD	Record Type Definition (Definición de tipo de registro)
RTN	Record Type Name (Nombre de Tipo de Registro)
SCK	Serial Clock (Reloj serie)
SD	Secure Digital (Seguro Digital.)
SDK	Software Development Kit (Kit de Desarrollo de Software)
SE	Secure Element (Elemento Seguro)
SIM	Subscriber Identify Module (Modulo de Identificación de Abonado)
SNEP	Simple NDEF Exchange Protocol (Protocolo de Intercambio Simple NDEF)
SPI	Serial Peripheral Interface (Protocolo de Interfaz Serial)
SR	Short Record (Registro corto)
SRAM	Static Random Access Memory (Memoria Estática de Acceso Aleatorio)
SS	Slave Select (Selecciodo de dispositivos)
TNF	Type Name Format (Nombre del tipo de formato)
UART	Universal Asynchronous Receiver-Transmitter (Transmisor-Receptor Asíncrono Universal)
URI	Uniform Resource Identifier (Identificador de Recursos Uniformes)
URL	Uniform Resource Locator (Identificador de Recursos Uniformes)
USA	United States of America (Estados Unidos de América)
USB	Universal Serial Bus (Bus Universal en Serie)
VCD	Vicinity Coupling Device (Dispositivo de Acoplamiento de Proximidades)
XML	Extensible Markup Language (Lenguaje de Marcas Extensible)

RESUMEN

Se analizó, diseñó e implementó un sistema de comunicación utilizando la tecnología inalámbrica de campo cercano (NFC), para el acceso a la información académica de los estudiantes de la Facultad de Informática y Electrónica FIE - ESPOCH. Se estableció un análisis comparativo entre varias alternativas de selección de tecnologías idóneas y placas Arduino para luego ser ocupadas en la implementación. El canal de comunicación inalámbrico entre los dispositivos utilizó principalmente los protocolos NDEF y LLCP definidos sobre los estándares ISO/IEC 18092 y el del Instituto de Ingenieros Eléctricos y Electrónicos (IEEE 802.2). En el diseño y posterior implementación se requirió una placa Arduino Mega 2560 que actuó como centro de control y procesamiento, ayudando como intermediario entre los dispositivos: Shield Ethernet que se encargó de enviar y recibir las tramas IP requeridas y un shield Adafruit PN532 para la transmisión y/o recepción de la petición y posterior envío de la solicitud desde la base de datos en forma inalámbrica al smartphone; además se contó con una aplicación móvil que posee una lista detallada de todos los documentos y/o archivos teniendo asignado un número de identificación que fue enviado posteriormente a través de la tecnología hacia el shield NFC, siendo receptado y procesado por el dispositivo de control, el cual realizó la petición respectiva al servidor de archivos desarrollado en el Sistema de Base de Datos Operacional (MySQL) y desplegada en una PC para su verificación y almacenamiento. Los resultados obtenidos indicaron que el sistema posee un alto porcentaje de eficiencia de 72.84% comparado con el método tradicional. Gracias al desarrollo de este sistema se logró agilitar los procesos empleados en cada trámite realizado, obteniendo una descarga segura y fiable en tiempo real. La instalación de este tipo de sistema y dispositivos es muy recomendable en instituciones de educación debido a la gran ayuda, eficacia y costo.

Palabras claves: <TECNOLOGIA Y CIENCIAS DE LA INGENIERÍA>, <TECNOLOGÍA DE COMUNICACIONES>, <COMUNICACIÓN DE CAMPO CERCANO (NFC)>, <COMUNICACIÓN INALÁMBRICA>, <PLATAFORMA ARDUINO>, <SOFTWARE ANDROID STUDIO>, <SISTEMA DE BASE DE DATOS OPERACIONAL (MySQL)>,

ABSTRACT

It was analyzed, implemented or designed a communication system using the near-field wireless technology (NFC), for access to academic information of students of the Faculty of Computer Science and Electronics FIE - ESPOCH. A comparative analysis of various alternatives and selection of best Arduino boards and then be engaged in the implementation technologies was established. The wireless communication channel between the devices primarily used the NDEF and LLCP protocols defined on ISO / IEC 18092 standards and the Institute of Electrical and Electronics Engineers (IEEE 802.2). Shield Ethernet that was responsible for sending and receiving IP frames required and Shield Adafruit PN532 for transmission: the design and subsequent implementation of an Arduino Mega 2560 that acted as a control center and processing, helping as an intermediary between the devices required and / or receipt of the request and subsequent submission of the application from the database to the Smartphone wirelessly; in addition he had a mobile application that contains a detailed list of all documents and / or files having assigned an identification number which was subsequently sent through technology to the shield NFC, being receipted and processed by the control device, which he made the respective request to the file server developed in the System Operational Database (MySQL) and displayed on a PC for verification and storage. The results indicated that the system has a high effectiveness rate of 72.84% compared to the traditional method. Thanks to the development of this system it was possible to expedite the processes used in each procedure performed, obtaining a secure and reliable real-time download. The installation of this Type of system and devices is highly recommended in educational institutions because of the great help, effectiveness and cost.

Keywords: <TECHNOLOGY AND ENGINEERING SCIENCES>, <COMMUNICATIONS TECHNOLOGY>, <COMMUNICATION FIELD NEAR (NFC), <WIRELESS COMMUNICATION, <PLATFORM ARDUINO, <SOFTWARE ANDROID STUDIO, <SYSTEM OPERATING DATABASE (MySQL).

INTRODUCCIÓN

El presente trabajo trata de la combinación de diversas tecnologías y softwares en un sistema único capaz de proporcionar información académica de relevancia para los estudiantes de la FIE-ESPOCH.

En este sistema se utilizará redes inalámbricas las cuales utilizan el espectro electromagnético a través de sus ondas para enviar y recibir la información, este procedimiento será utilizado para establecer la comunicación entre el módulo NFC.

El sistema de comunicación desarrollado básicamente constará de tres partes fundamentales: La primera será crear la aplicación para el sistema de gestión y visualización de la información hacia el usuario final, la segunda será establecer el canal de comunicación que se encargará de la transmisión y/o recepción de información entre el teléfono móvil con la base datos y la tercera es desarrollar en el servidor la base de datos donde se almacenará la información.

A continuación se expondrá la importancia de las comunicaciones inalámbrica y con ello la tecnología NFC, sus características, desarrollo, impacto y aplicación en el mundo, así como también la utilización y desarrollo de herramientas de hardware y software con sus distintas plataformas como son Arduino y Android, y la importancia del desarrollo de un sistema de comunicación mediante NFC para el acceso a la información académica de los estudiantes, las necesidades que cubrirá y los beneficios que proporcionará.

Además se expone los objetivos, métodos y técnicas a seguir para el desarrollo y culminación de este trabajo de titulación.

ANTECEDENTES DEL PROBLEMA

El término NFC apareció por primera vez en el año 2002 con el propósito de intentar conseguir un protocolo compatible con las tecnologías para la transmisión sin contacto existentes en ese momento y con el fin de obtener una tecnología de corto alcance que permita una conexión instantánea entre dispositivos y establecer una comunicación de manera intuitiva, sencilla, ágil y simple.

NFC es una tecnología inalámbrica de corto alcance óptima para una comunicación instantánea, siendo capaz de enviar y recibir información al mismo tiempo, uno de los principales usos es la identificación y validación de equipos/personas.

NFC entró con fuerza en la industria de las comunicaciones móviles hace unos años y, aunque parecía ser una alternativa a Bluetooth e IrDa, muy pocos fabricantes fueron los que se lanzaron al mercado de NFC, lo que provocó que dicha tecnología no terminara de despegar, sin embargo en la actualidad muchos dispositivos ya integran esta tecnología de forma nativa.

Los últimos años han sido testigos del creciente interés del uso potencial de las comunicaciones inalámbricas y sus diversas aplicaciones, hoy en día están inmersas en las actividades cotidianas, ayudando sin duda a mejorar la calidad o estilo de vida de las personas.

La tendencia de crecimiento y expansión de NFC en el mundo es evidente, más aún en empresas de telefonía móvil donde integran esta tecnología en la mayoría de sus dispositivos, brindándonos un sin número de beneficios como compartir contactos, fotos, videos o archivos, además del potencial de los dispositivos con NFC para que actúen como documentos de identidad electrónicos y keycards.

En la actualidad la tecnología NFC muestra un gran impacto tecnológico y social en el mundo entero sobre todo en los sistemas de comunicación, lo que la convierte en una herramienta ideal que forma parte de las actividades cotidianas.

En esta década NFC ha tenido un gran desarrollo, abriendo un amplio abanico de posibilidades y aplicaciones atractivas para el usuario, ya que ayuda a automatizar muchas funciones o actividades de la vida diaria contribuyendo a hacerla mucho más sencilla, pudiéndola ocupar en diferentes aplicaciones dependiendo de las necesidades de los usuarios.

El Software y Hardware Libre, se caracterizan por su fácil acceso y manejo, sin estar atados a ningún tipo de licencia o patente para su uso, dando grandes prestaciones y siendo muy versátil en el desarrollo de nuevas aplicaciones y usos, permitiendo a su vez que comunidades que trabajan sobre estos aporten valiosas ideas que ayudan al desarrollo de las distintas tecnologías, siendo las plataformas Arduino y Android los ejemplos más representativos.

La Facultad de Informática y Electrónica de la Escuela Superior Politécnica de Chimborazo forma parte de las entidades de educación superior que tiene un alto prestigio en la formación integral de profesionales líderes, honestos y preparados, con el objetivo de contribuir al desarrollo científico, técnico, social, económico e industrial del país.

JUSTIFICACIÓN DEL TRABAJO DE TITULACIÓN

A continuación se detalla la justificación teórica y aplicativa del sistema a desarrollar:

JUSTIFICACIÓN TEÓRICA

Las entidades educativas y/o personas buscan dinamizar, optimizar y reducir los tiempos de procesos o trámites, con el fin de hacer más fácil o sencilla la actividad realizada y disminuir el tiempo empleado, como respuesta a ello nace el uso de nuevas tecnologías con el fin de satisfacer los requerimientos y necesidades de los usuarios finales, tal es el caso de la tecnología NFC que mediante ondas electromagnéticas permite transmitir y recibir datos de forma inmediata.

Se desarrollará un sistema de comunicación basado en la tecnología NFC con el propósito de proveer un mecanismo de información que permita optimizar los tiempos de respuesta a los procesos, dentro de ello se utilizará dos plataformas flexibles como son Arduino para el procesamiento de los datos y Android para la interfaz de comunicación con el usuario, basadas en software y hardware libre, permitiendo así reducir costos y obtener mayores beneficios.

Para el desarrollo de la aplicación se empleará SDK Tools y Android Studio, que son herramientas de desarrollo de aplicaciones para Android y JDK que es un software que provee herramientas para la creación de programas en Java, esta interfaz permitirá que el usuario envíe las peticiones mediante NFC y posteriormente el Arduino mande la información almacenada en el servidor, dando como resultado que el usuario final obtenga el requerimiento enviado en la petición inicial.

Para el almacenamiento de la información académica se ocupará el gestor de base de datos MySQL, el cual contará con la información requerida como es el caso de oficios o solicitudes previas a cualquier trámite.

En la comunicación y la realización de las consultas utilizaremos Apache conjuntamente con PHP que son herramientas web y servirán de intermediarios e intérpretes de las peticiones realizadas desde la placa reducida Arduino hacia la base de datos.

Como resultado final se logrará desarrollar un sistema de comunicación con NFC que proporcionará a los estudiantes los documentos necesarios como modelos de solicitudes para

diversos trámites, mallas curriculares, entre otros, facilitando y simplificando de esta manera los procesos internos de la facultad.

JUSTIFICACION APLICATIVA

En el transcurso del ciclo académico existe un gran entorpecimiento de los procesos rutinarios de la facultad, debido a la falta de información y acceso a documentación de vital importancia para los estudiantes como son: solicitudes y oficios, lo que genera grandes colas y con ello no se dinamizan los trámites correspondientes.

El desarrollo de un sistema de comunicación con NFC para el acceso a información académica de los estudiantes de la FIE-ESPOCH ayudará a reducir este problema ya que permitirá obtener información de manera fácil, ágil e intuitiva mediante dispositivos móviles que posean esta tecnología y un lector NFC que tendrán un enlace directo a los archivos almacenados en una base de datos implementada en un servidor Open Source

El sistema de comunicación propuesto evidentemente agilitará y mejorará los procesos, beneficiando así a los estudiantes que necesiten información específica de la facultad y ayudando a reducir el tiempo empleado para los trámites de cada usuario.

El sistema tendrá la capacidad de permitir el acceso a la siguiente información de la Facultad de Informática y Electrónica:

- Formatos de Solicitudes (Oficios).
- Croquis de las instalaciones.
- Datos Informativos: Autoridades y Personal Administrativo
- Galería de la Facultad.

FORMULACIÓN DEL PROBLEMA

¿En qué nivel mejorará el tiempo de acceso a la información académica por parte de los estudiantes de la FIE-ESPOCH y cuál será el tiempo promedio de descarga de dicha información con la tecnología NFC?

METODOLOGÍA

Este proyecto tecnológico requiere un estudio a través de la aplicación de métodos de investigación científica, para lo cual se aborda la investigación documental, mediante la cual se investiga el estado del arte de NFC a través de la recopilación de documentos que contengan información relevante de esta tecnología, lo relacionado a Arduino y Android como conceptos teóricos o de investigación pura con el fin de obtener conocimiento útil.
La metodología se basa en tres actividades para el desarrollo del trabajo de titulación, donde se ocupa deductivo, sintético, deductivo y de análisis, que se especifican a continuación:

Actividad 1: Para la investigación del funcionamiento de NFC se utiliza el método inductivo y deductivo para analizar y comparar su comportamiento, con el fin de obtener los mejores beneficios que brinda la misma.

Actividad 2: Para el diseño del sistema de gestión académica se utiliza el método deductivo y de análisis para ver los diferentes tipos de hardware y software que existen, con ello poder optar por el diseño que mejor cubra los requerimientos.

Actividad 3: Para lograr la realización del sistema de información académica se utiliza el método analítico, por medio del cual se obtendrá una instalación correcta de cada uno de los elementos, logrando así no tener problemas en su funcionamiento.

OBJETIVOS

Objetivo General

- Desarrollar un sistema de comunicación con NFC para el acceso a información académica de los estudiantes de la FIE-ESPOCH.

Objetivos Específicos

- Estudiar el estado del arte de la tecnología NFC y sus aplicaciones en el ámbito de la gestión académica.
- Implementar el hardware del sistema de comunicación con NFC.
- Desarrollar la interfaz de usuario para el acceso a la información académica
- Verificar el adecuado funcionamiento del sistema implementado.

CAPÍTULO I

1 MARCO TEÓRICO REFERENCIAL

1.1 Introducción

Las comunicaciones surgen de la necesidad del ser humano de estar en contacto de forma constante, a tal punto que a través del tiempo da origen al desarrollo tecnológico y conlleva a la aparición de medios de comunicación eficaces, gracias a ello los obstáculos que afrontan la sociedad día a día son resueltos con soluciones tecnológicas.

Actualmente la tecnología crece cada vez más gracias a los avances de los diferentes tipos de comunicaciones inalámbricas en todo el mundo, este trabajo estudia una de esas tecnologías que nace con el propósito de cubrir nuevas necesidades, resolver problemas y facilitar la vida de las personas, tal es el caso de NFC que gracias a sus diversas aplicaciones hacen que las actividades de la vida cotidiana sean más sencilla a través de un simple toque.

En el desarrollo de este capítulo se pretende conocer información relacionada a la tecnología NFC, tarjetas de desarrollo y Android, así como identificar los beneficios y características de cada una de ellas.

1.2 Near Field Communication (NFC)

Es la comunicación de campo cercano, una tecnología de transmisión inalámbrica de corto alcance que se comunica vía electromagnética utilizando los principios básicos de RFID, que fue su tecnología predecesora, se dice que es la unión de esta con otras tecnologías interconectadas, permitiendo el intercambio de datos de manera instantánea de forma unidireccional o bidireccional a una distancia menor a 10 cm.

Es una prolongación del estándar ISO/IEC 14443 de tarjetas inteligentes de proximidad, que combina la interfaz de la tarjeta de identificación electrónica sin contacto y un lector incorporado dentro del dispositivo, la mayoría de veces utiliza una comunicación bidireccional o half-duplex, es decir en ambos sentidos pero no al mismo tiempo.

Es una tecnología abierta, que fue diseñada principalmente para su utilización en dispositivos móviles, capaz de establecer comunicación fácilmente con otros dispositivos NFC u otras infraestructuras inalámbricas existentes, trabaja a una frecuencia de 13.56 Mhz, disponible sin restricción alguna y sin necesidad de licencia o permiso para su uso, su tasa de transferencia llega a 424 kbit/s, donde la idea no es transmitir grandes cantidades o volúmenes de datos, sino intercambiar información de manera sencilla, rápida, ágil y segura.

Los sucesos que contribuyeron en el desarrollo de NFC a partir del año 1940 hasta el 2010 se especifican en la siguiente tabla:

PERÍODOS	SUCESOS Y AVANCES
1940-1950	Desarrollo de sistemas de identificación IFF y dispositivos de escucha pasivos usados en la segunda Guerra mundial.
1950-1960	Indagaciones iniciales de RFID, desarrollo del transpondedor de largo alcance IFF.
1960-1970	Comienzo de aplicaciones RFID, primer sistema para vigilancia electrónica de articulos llamado EAS.
1970-1980	Notables avances, comienzo de la explotación de RFID y sus primeras patentes.
1980-1990	Implementación masiva de RFID y desarrollo en España como identificación para el ganado en sector privado.
1990-2000	RFID toma relevancia e importancia en la vida cotidiana, debido a su bajo coste gracias a IBM que integra todo el circuito en un solo chip.
2000-2010	Nacimiento de la tecnologia NFC con cambios y mejoras de su predecesora RFID e implementación en el mercado, creación de aplicaciones en el mercado que mejoran las actividades diarias y aceptación en aumento.

Figura 1-1: Sucesos para el desarrollo de NFC
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

1.2.1 Roles de NFC en la comunicación

Un sistema de comunicación tiene por objetivo primordial transmitir información desde la fuente al destino a través del canal de transmisión, en el caso de NFC al ser una comunicación inalámbrica utiliza el espectro electromagnético para la modulación de sus ondas, para cumplir con este proceso un dispositivo NFC puede tomar dos roles:

1.2.1.1 Rol Iniciador

Se le considera al dispositivo que empieza la interacción o sesión enviando la petición inicial, es decir el emisor de la comunicación, generando el campo de radiofrecuencia que será aprovechado por el objetivo.

1.2.1.2 Rol Objetivo

Se le considera dispositivo objetivo a aquel que responde la comunicación generada por el iniciador enviando un mensaje de retorno, es decir el receptor de la comunicación.

1.2.2 Establecimiento de la conexión

Toda comunicación NFC se realiza en las siguientes etapas:

- **Descubrimiento:** Los dispositivos se rastrean para poder reconocerse.
- **Autenticación:** Etapa de verificación con el fin de saber los dispositivo están autorizado o si deben establecer algún tipo de cifrado para establecer una comunicación segura.
- **Negociación:** Definen parámetros de funcionamiento para la transmisión y la acción a ser solicitada.
- **Transferencia:** Etapa de intercambio o interconexión de la información.
- **Confirmación:** El dispositivo receptor realiza la confirmación del establecimiento de la comunicación y la transferencia de datos.

1.2.3 Interacción NFC

Es sumamente sencilla e intuitiva, no requiere de configuraciones adicionales, es tan fácil como tener cerca los dos dispositivos NFC, esperar que se reconozcan y realizar el toque para la transferencia de datos.

La Figura 2-1 muestra el acoplamiento automático de un dispositivo móvil y su objetivo que puede ser un tag, móvil o lector, este proceso inicia al momento que el usuario realice la interacción respectiva.

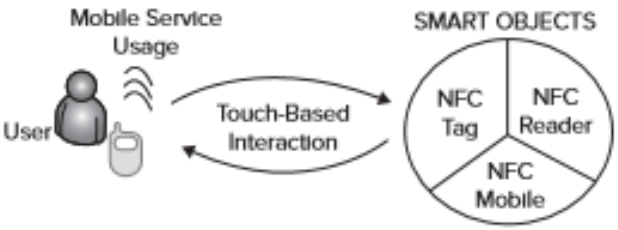


Figura 2-1: Esquema de interacción NFC
Fuente: Coskun, Ok & Ozdenizci, 2013.

1.2.4 Especificaciones Técnicas

NFC transmite pequeñas cantidades de datos al acercar el terminal al receptor a una alta frecuencia, diseñada para una comunicación instantánea, de uso transparente a los usuarios que funciona en la banda libre de 13,56 MHz.

La distancia de trabajo de NFC es generalmente efectivo cuando se está lo más cercano a 10 cm, tomando menos de 0.2 segundos en establecer la comunicación y soporta velocidades de intercambio de datos o transmisión de 106, 212,424 u 848 kbits/s dependiendo de sus modos de comunicación. El consumo de energía de un dispositivo en la lectura de datos es menor de 15mA, pudiendo ser mayor en la escritura de datos.

NFC obedece a protocolos como:

- **LLCP:** Protocolo compacto OSI de nivel 2 de control lógico de link y orientado para conexiones que es usado para la activación, supervisión y desactivación de la comunicación proveyendo un enlace entre dos dispositivos activos habilitados con esta tecnología para las comunicaciones bidireccionales o modo de operación punto a punto.

- **NDEF:** Proporciona un formato para el intercambio fiable de datos, definiendo una modalidad de formateado/encapsulamiento internamente en el mensaje en conjunto con LLCP.
- **SNEP:** Protocolo de intercambio simple de NDEF orientado para mensajes.
- **RTD:** Es un formato optimizado para la transmisión entre dispositivos NFC.

En la figura 3-1 se aprecia la arquitectura tecnológica de NFC y detalla los protocolos utilizados en cada uno de los modos de comunicación

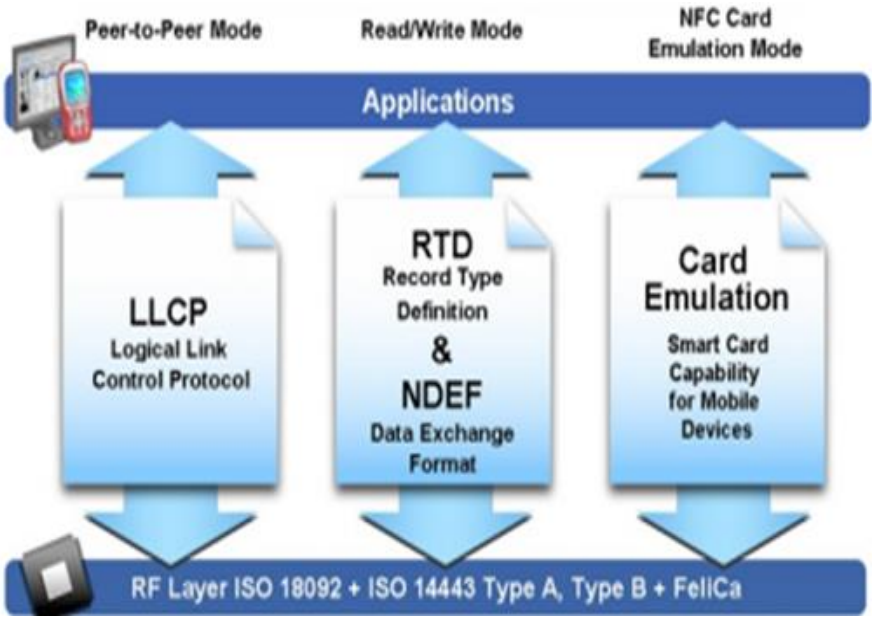


Figura 3-1: Arquitectura tecnológica de NFC
Fuente: http://histinf.blogs.upv.es/files/2012/11/HDI-Trabajo_NFC.pdf

1.2.4.1 Estándares de Regulación

NFC está reconocido por los organismos internacionales ISO/IEC, ETSI y ECMA, los estándares de regulación de esta tecnología son especificaciones técnicas de protocolos de comunicación o formatos de intercambio de datos y se basa en dos normas internacionales que son:

Norma ISO/IEC 18092/ECMA-340/ETSI TS 102 190

NFC fue estandarizada por la norma ISO/IEC 18092 en diciembre del 2003, y posteriormente como ECMA-340.

Define los modos de comunicación de campo cercano activo y pasivo con sus rangos de velocidad de (106-424) kbps para la interfaz NFC and Protocol-1 (NFCIP-1), esquemas de modulación, codificación e inicialización, velocidad de transferencia y mecanismos de control, utilizando dispositivos de acoplamiento inductivo.

Norma ISO/IEC 21481/ECMA-352/ETSI TS 102 312

Pertenece a Near Field Communication Interface and protocol-2 (NFCIP-2), esta norma determina mecanismos para seleccionar el adecuado modo de operación al inicio de la comunicación.

Los modos de operación pueden ser NFC de ECMA-340, PICC de ISO/IEC14443, PCD de ISO/IEC 14443 y VCD de ISO/IEC 15693, diferenciándose entre ellos por la distancia de detección del campo de radiofrecuencia, los procedimientos de inicialización y los valores mínimos de sus campos para detección, previniendo de esta manera interferencias con cualquier comunicación en curso de la banda 13,56 MHz.

1.2.5 Modos de Funcionamiento

Los dispositivos que poseen la tecnología NFC para la transmisión y/o recepción de datos pueden trabajar u operar bajo dos modos de funcionamiento diferentes, estos son:

1.2.5.1 Modo de funcionamiento Pasivo

Se produce cuando un dispositivo no propaga un campo propio, sino que recibe alimentación del campo electromagnético del dispositivo iniciador de la comunicación, utilizando esta modulación para transferir o intercambiar datos, por esta circunstancia siempre será el objetivo de la comunicación y nunca podrá ser el iniciador ya que por sí sólo es incapaz de emitir la señal hacia otro dispositivo.

1.2.5.2 Modo de funcionamiento Activo

Se produce cuando los dos dispositivos emisor y receptor generan sus propios campos electromagnéticos o radiofrecuencia, generando la portadora que ayuda a establecer el canal de comunicación, pudiendo actuar como iniciador u objetivo de la comunicación.

1.2.6 Modos de Comunicación u Operación

NFC puede trabajar bajo tres modos de comunicación u operación que se detallan a continuación:

1.2.6.1 Modo Punto a Punto

Permite que dos dispositivos habilitados puedan comunicarse alternando la generación de sus campos electromagnéticos y a través de un solo toque, para intercambiar y compartir datos, cuya velocidad aproximada es 424 Kbit/s.

La figura 4-1 establece el modo punto a punto o también conocido por sus siglas en ingles Peer to peer (P2P), donde los dispositivos NFC crean la conexión de forma activa estableciendo la comunicación en forma bidireccional half-duplex o uno en ese momento, es decir que cuando uno transmite el otro escucha e inicia su proceso cuando el otro haya terminado, valiéndose de una aplicación o servicio integrado dentro del móvil.

Figura 4-1: Uso peer to peer de dispositivos NFC
Fuente: http://histinf.blogs.upv.es/files/2012/11/HDI-Trabajo_NFC.pdf

1.2.6.2 Modo Lectura-escritura

Este modelo es dedicado a la lectura- escritura de la información almacenada en las etiquetas NFC como se muestra en la figura 5-1, las cuales actúan como dispositivos pasivos, donde la velocidad de transmisión es aproximadamente 106 Kbit/s.

Figura 5-1: Uso lectura-escritura de dispositivos

Fuente: http://histinf.blogs.upv.es/files/2012/11/HDI-Trabajo_NFC.pdf

1.2.6.3 Modo Emulación de tarjeta

Este modelo permite a los dispositivos NFC emular o actuar como una tarjeta inteligente, el cual se comunica con un lector externo donde se puede obtener información como se aprecia en la Figura 6-1, utiliza un protocolo propio y funciones especiales de seguridad lo que lo hace mejor que un tag normal, manteniendo su funcionalidad a pesar que el dispositivo este apagado dependiendo del proveedor de servicios que posea.

Figura 6-1: Uso emulación de tarjeta de dispositivos NFC
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

1.2.7 NFC Forum

Es una entidad oficial creada en el año 2004 por empresas líderes de comunicaciones móviles en ese momento, es la encargada de regular, determinar las características, estándares, interacción e interoperabilidad de sus dispositivos y promover NFC en el mundo entero.
Los miembros que pertenecen a esta organización logran establecer la confianza, credibilidad y la más amplia garantía en sus productos, debido a que NFC Forum proporciona fiabilidad e integridad y garantiza el más alto nivel de calidad en todos los productos, procesos o servicios; antes de ser aprobado pasa por un estricto proceso de revisión para verificar que las aplicaciones cumplan con las políticas, los requerimientos técnicos garantizando de esta manera la eficacia, la eficiencia de su certificación y asegurando la interoperabilidad entre fabricantes.

Los objetivos que cumple esta organización son los siguientes:

- Desarrollar especificaciones y mecanismos de prueba de tal manera que garanticen transacciones consistentes y confiables en todo el mundo en los tres modos de NFC.
- Tomar un papel de liderazgo en la industria para asegurar que la tecnología NFC puede proporcionar de forma rutinaria una experiencia de usuario única y positiva.
- Educar a las empresas, proveedores de servicios y desarrolladores acerca de los beneficios de la tecnología NFC para asegurar el excelente crecimiento en la adopción del usuario final.
- Establecer NFC FORUM y las marcas de la tecnología NFC como únicas marcas reconocidas y utilizadas.

1.2.8 Formato de Intercambio de Datos (NDEF)

Es un formato binario ligero normalizado de intercambio y almacenamiento de información, diseñado por NFC Forum con el propósito de que dispositivos NFC puedan compartir datos sin ninguna dificultad.

Se usa para identificar el tipo de dato aplicativo, se encarga de guardar y transportar diferentes tipos de elementos, para ello define un formato de encapsulación de mensajes ya sea de un dispositivo o una etiqueta NFC, garantizando así la coexistencia e interoperabilidad entre los distintos dispositivos y aplicaciones.

Se convierte en un lenguaje común entendido por ambas partes de la comunicación NFC ya sean etiquetas o dos dispositivos activos en el modo punto a punto y ese utiliza para compartir distintos tipos de datos de forma organizada y comprensible, a continuación se presenta las especificaciones:

1.2.8.1 Estructura de Registro NDEF

Posee un formato simple y común, con longitud variable y su información se representa a nivel de octetos, donde la transmisión se ejecuta desde el bit más significativo el cual se transmite primero que es el del extremo izquierdo hacia el derecho y de la parte de superior a la inferior.

Un registro está compuesto por una cabecera que posee información propia necesaria para leer los datos y por el payload que comprende el contenido del mensaje que se está transmitiendo.

En la Figura 7-1 se establece el formato de un registro NDEF, cuyas descripciones se detallan en la Tabla 1-1 realizada a continuación.

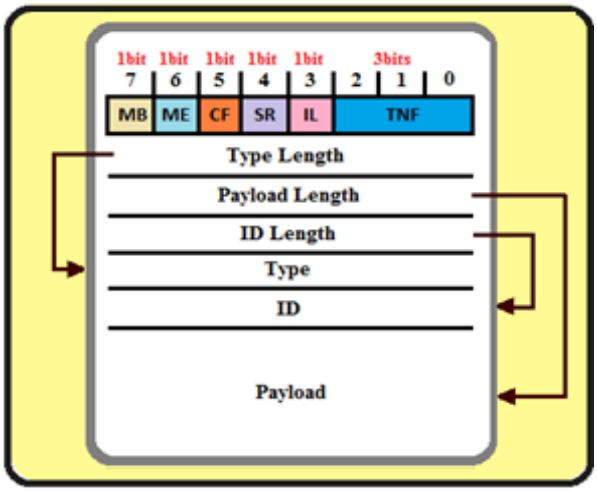


Figura 7-1: Formato de un registro NDEF
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

Tabla 1-1: Descripción de las partes de un registro NDEF

Indicador	Descripción	Tamaño
MB	Especifica al comienzo de un mensaje NDEF	1 bit
ME	Determina la terminación de un mensaje NDEF	1 bit
CF	Da a conocer si es el trozo de la mitad o el primer trozo del registro de un payload fragmentado.	1 bit
SR	Indica que el campo PAYLOAD_LENGT está constituido no de cuatro octetos como uno registro normal sino por uno sólo.	1 bit
IL	Especifica que el campo ID_LENGTH está activo como un octeto en la cabecera del registro, pero si IL es cero es omitido por la cabecera y el campo ID también es omitido del registro.	1 bit
TNF	Indica la estructura del valor de campo TYPE los cuales pueden ser de varios tipos como 0x00, 0x01, 0x02, 0x03, 0x04, 0x05, 0x06, 0x07 e indica cómo se interpreta el mensaje.	3 bit
TYPE_LENGTH	Identifica el formato del registro, entero no asignado que especifica la longitud de octetos del campo Type.	8 bits
PAYLOAD_LENGTH	Entero no asignado que obedece al valor de la bandera SR, si está establecida representa un solo octeto pero si está vacía es de 4 octetos.	8 bits 32 bits
ID_LENGTH	Entero no asignado activo IL está en 1 y da en octetos la longitud del campo ID	8 bits
TYPE	Define el tipo de payload de la información transmitida.	Max 255 octetos
ID	Posee el valor de una referencia URI o una cadena de texto que identifica un nombre, localización o característica de un recurso determinado.	Max 255 octetos
PAYLOAD	Lleva la carga destinada para aplicaciones de usuario NDEF.	-

Fuente: http://bibdigital.epn.edu.ec/bitstream/15000/2227/1/CD-2970.pdf
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

En la Tabla 2-1 se especifica los valores que puede tomar el indicador TNF:

Tabla 2-1: Valores del campo TNF

Tipo	Valor	Descripción
Vacío	0x00	No hay ningún tipo o payload asociado al registro.
Well-known NFC Forum	0x01	Indica que el campo Type contiene un valor que sigue el formato de RTD.
MIME Tipo de medio	0x02	Obedece a un valor del formato BNF definido en RFC 2046.
URI absoluto	0x03	Sigue a un valor de construcción absoluta URI-BNF definido en RFC 3986.
NFC Forum Externo	0x04	Valor para nombres de tipo externo definido en NFC IDT
Desconocido	0x05	Campo de payload desconocido
Sin cambio	0x06	No debe usarse en otro registro que no sean los fragmentos de registro de medio y terminal que forman payloads segmentadas
Reservado	0x07	Para usos futuros, no debe ser usado.

Fuente: http://www.eet-china.com/ARTICLES/2006AUG/PDF/NFCForum-TS-NDEF.pdf
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

1.2.8.2 Mensajes NDEF

Un mensaje NDEF es un mecanismo de transporte básico elemental para registros NDEF y en su estructura puede contener uno o más registros ilimitados marcados al inicio por MB y al final lleva la bandera ME y encapsula uno o más cargas llamadas payloads de diferente tipo y tamaño, sin embargo por convención el tipo del primer registro determina como se procesa el mensaje por completo. La cabecera está conformada por 8 bits, donde los primeros cinco bits tienen información sobre como procesar y la ubicación del registro en el mensaje, este proceso se visualiza en la Figura 8-1 dada a continuación:

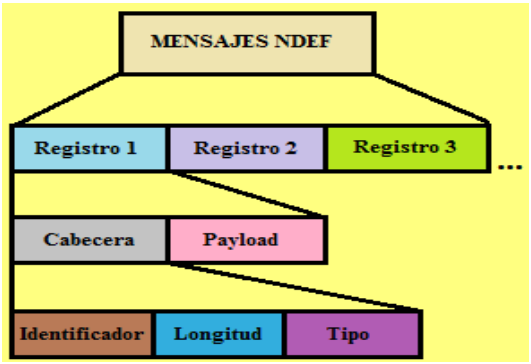


Figura 8-1: Estructura de un mensaje NDEF
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

1.2.8.3 Fragmento de Registros NDEF

En el mensaje NDEF un registro es la unidad para transportar un payload o carga útil que se encuentra representada y descrita por sus propios parámetros.

En ocasiones con el fin de particionar un contenido o mensajes demasiado largos se pueden subdividir o fragmentar en múltiples pedazos de registros ordenados en un mismo mensaje NDEF.

Las reglas para la fragmentación de payloads son:

Los payload fragmentados en dividen en tres pedazos de registro inicial, medio y final que son codificados cumpliendo reglas específicas.

1. Para el fragmento de registro inicial:

- Debe contener la bandera CF (Chunk Flag) que señala que se trata del primer trozo del registro.
- En el campo TYPE debe estar indicado el tipo de toda la payload fragmentada.
- Para llevar un identificador de toda la payload fragmentada se podría utilizar el campo ID.
- En campo longitud de payload se debe especificar el tamaño de datos que lleva el registro actual y no del payload completo.

2. Para el fragmento de registro medio:

- Posee la bandera CF señalando que este registro se trata del trozo intermedio o la próxima porción de datos de igual tipo e identificador como el fragmento inicial.
- El valor de campo TYPE_LENGTH y de ID_LENGTH debe ser cero y el campo TNF debe ser 0x06.

3. Para el fragmento de Registro final:

- Debe tener la bandera CF limpia señalando que se trata del último fragmento o porción de datos de igual tipo e identificador que el fragmento inicial. El valor de campo TYPE_LENGTH y de ID_LENGTH debe ser cero y el campo TNF debe ser 0x06.

4. Un payload fragmentado nunca puede contener múltiples mensajes NDEF sino que debe ser completamente encapsulado en un sencillo mensaje NDEF, por lo que en ningún caso el fragmento de registro inicial o de registro medio deben tener una bandera ME establecida.

1.2.8.4 Definición de Tipo de Registro (RTD)

La especificación RTD proporciona un formato optimizado que puede ser incluido en mensajes NDEF transmitidos entre dispositivos o tags NFC, permitiendo soportar aplicaciones específicas NFC y especifica los tipos de registros estándar que pueden ser enviados en mensajes.

Cada RTD es reconocida por su RTN (Record Type Name) que a su vez estos son especificados en los diferentes formatos de TNF (Type Name Format), los tipos de formatos RTD son:

- ***Texto RTD:*** Utilizado para registros que contienen exclusivamente solo datos.
- ***Smart Poster RTD:*** Son elementos adheridos empleado para posters que incorporan etiquetas con datos.
- ***Uniform Resource Identifier (URI) RTD:*** Usado para registros que sugieren a recursos en internet.

1.2.9 Modulación y Codificación NFC

Para la transmisión de su información utiliza la Modulación digital por desplazamiento de amplitud (ASK) con profundidades del 10 o 100%.

La modulación por desplazamiento de amplitud es un método mediante el cual se modifica o varia la amplitud de la señal portadora de acuerdo a la moduladora que es la información digital, donde los dos valores binarios de los datos de entrada son representados por dos diferentes amplitudes de la frecuencia portadora.

Utiliza la codificación Manchester y Miller dependiendo de sus modos de funcionamiento y velocidad de transmisión, debido a que la información digital al no poder ser transmitida como una señal binaria que es en forma de 0 y 1 debe ser codificada en una señal con dos estados para que pueda ser correctamente enviada a través del medio de comunicación en forma de señales digitales.

La codificación Manchester es también conocida como codificación de 2 fases o bifase-L, facilita un extraordinario sincronismo entre el receptor y transmisor, ya que es un método auto sincronizado por medio del cual cada período o tiempo de bit tiene una transición en la parte media de cada intervalo.

La codificación Miller se la conoce también como retrasada y es muy parecida a la codificación Manchester pero tiene una variación que permite tener un mayor índice de datos, esta codificación es utilizada en el modo activo de NFC en una velocidad de 106 kbps.

A continuación se detalla el tipo de modulación y codificación NFC en sus dos modos activo y pasivo dependiendo de su velocidad:

Tabla 3-1: Tipo de modulación y codificaciones NFC

Velocidad	Modo Activo		Modo Pasivo	
	Modulación	***Codificación***	***Modulación***	***Codificación***
106 kbps	ASK al 100% Índice de modulación: 1	Miller	ASK al 10% Índice de modulación: 0.1	Manchester
212 kbps	ASK al 10% Índice de modulación: 0.1	Manchester	ASK al 10% Índice de modulación: 0.1	Manchester
424 kbps	ASK al 10% Índice de modulación: 0.1	Manchester	ASK al 10% Índice de modulación: 0.1	Manchester

Fuente: http://cwi.unik.no/images/Master_thesis_lu_NFC.pdf
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

1.2.10 Dispositivos Inteligentes NFC

Los dispositivos que cuentan con la tecnología NFC se encuentran en tres presentaciones que son teléfonos móviles, lectores y etiquetas.

1.2.10.1 Teléfonos Móviles

Un celular o Smartphone con esta tecnología se conforma básicamente por circuitos integrados, por elementos que proporcionen seguridad y por la interfaz NFC.

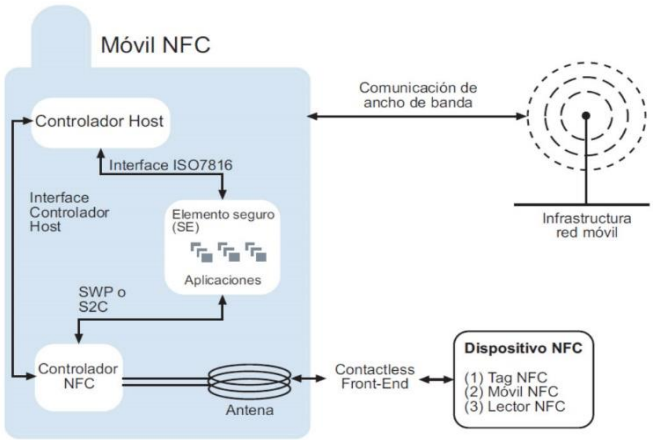


Figura 9-1: Arquitectura de un móvil NFC
Fuente: Coskun, Ok & Ozdenizci, 2013.

La Figura 9-1 muestra la arquitectura de un móvil que cuenta con la tecnología NFC, cuyas partes se detallan a continuación:

Controlador Host: Es el encargado de procesar los datos enviados y recibidos estableciendo la conexión entre el controlador NFC y el elemento seguro, por medio del HCI fija la modalidad operativa del controlador NFC, por lo que se considera como el corazón del móvil.

Elemento Seguro SE: Es una composición de hardware, software, protocolos e interfaces dentro del dispositivo, que permite procesar la transacción sin riesgos, ya que garantizan un almacenamiento seguro de información privada o confidencial gracias a la memorización y ejecución segura de datos y servicios, el cual puede ser accedido desde el interior por el controlador host o del exterior por el campo de radiofrecuencia.
En la actualidad un microcontrolador incorporado en el interior del dispositivo, una tarjeta de memoria o una tarjeta SIM, cualquiera de estos puede comportarse como elemento seguro, cada dispositivo tiene por lo menos un elemento de seguridad.

Host Controller Interface (HCI): Es una interfaz lógica que crea un enlace o puente enlace entre el controlador NFC y el controlador de Host, permitiendo el control y manejo del controlador NFC, posibilitando también la comunicación con los elementos seguros ya sean internos o externos.

Interfaz NFC: Está constituida por un contacto frontal analógico/digital o Contactless Front-End, una antena y por un controlador NFC que es un circuito integrado que posibilita efectuar las transacciones ya que trabaja actuando como modulador-demodulador entre la señal RF y la antena NFC, soportando los dos modos de funcionamiento y los tres modelos de operación o comunicación.

1.2.10.2 Lectores NFC

Es un dispositivo capaz de detectar, reconocer, leer y transferir la información de otro dispositivo con la misma tecnología y características similares como tarjetas inteligentes, dispositivos móviles y etiquetas sin contacto.

1.2.10.3 Etiquetas o Tags NFC

Son dispositivos NFC pasivos que permite realizar acciones configuradas dependiendo de las necesidades del usuario, y trabajan a través de un dispositivo activo que genera el campo electromagnético para que esta la aproveche, son de bajo costo y reducida capacidad.

A continuación se detallan los tipos de etiquetas o tag NFC y sus características:

Tabla 4-1: Tipos de etiquetas NFC y sus características

Tags	Tipo 1	Tipo 2	Tipo 3	Tipo 4
Interfaz RF	ISO 14443ª	ISO 14443ª	FeliCa 18092	ISO 14443 A y B
Capacidad Lectura/Escritura	SI	SI	Configurada de fabricación	Preconfigurada en fabricación
Permisos Usuarios	Solo lectura	solo lectura	Configuradas de fabricación lectura/escritura o sólo lectura	Configuradas de fabricación lectura/escritura o sólo lectura
Memoria	96 Bytes expandible 2Kb	48 Bytes expandible 2Kb	Variable, teóricamente límite es 1MB por servicio	32 KB por servicio
Velocidad de Comunicación	106 Kbps	106 Kbps	212 o 424 Kbps	424 Kbps
Costo	Bajo	Bajo	Alto	Medio/Alto
Anticolisión	No	Si	Si	Si

Fuente: http://nfc-forum.org/our-work/specifications-and-application-documents/specifications/tag-type-technical-specifications/
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

1.2.11 Tipos de Ataques a NFC

Por su corta distancia y proximidad con que realiza las operaciones es muy complicado que pueda ser atacada, sin embargo a pesar de su alto grado de seguridad tiene su punto débil y no está exenta presentando posibles vulnerabilidades como cualquier otra tecnología.

Existen 6 posibles tipos de ataques los cuales se mencionan a continuación:

Relay-Atack

Es un ataque que se utiliza para redirigir información de un dispositivo a otro, el cual se puede evitar realizando una doble autentificación o redundancia al realizar la actividad.

Denegación de Servicio

Este ataque consiste en enviar muchas peticiones al dispositivo, a tal punto que lo sature con el fin de impedir el acceso a los recursos o servicios durante un período indefinido de tiempo.

Intercepción o Sniffing

Es una técnica utilizada para escuchar o husmear la comunicación entre dos personas, es decir todo lo que sucede en una red, como datos importantes siempre y cuando no estén cifrados, es así que un atacante a través de una antena especial podría interceptar la comunicación entre dos dispositivos NFC, sin embargo es solo una posibilidad, ya que aún no existe una prueba contundente de la distancia exacta a la que el atacante debería ubicarse.

Una solución a este problema es debe tener varias capas de cifrado como posee en algunas de sus aplicaciones como es el caso del pago móvil, con ello el atacante a más de captar la información, tendría que poder descifrarla para comprender la comunicación.

Corrupción de Datos

Es un método que emplea el atacante con el fin de impedir u obstaculizar la comunicación, luego de interceptar los datos el siguiente paso es hacer la comunicación confusa, con el fin que sean imposibles entender por el lector o receptor de la información y provocando así una denegación de servicio.

Modificación e inserción de datos

La modificación no solo se limita a obstaculizar la comunicación sino que realiza cambios en datos originales con contenido válido el cual esta manipulado por el atacante, siendo necesario que conozca la codificación que emplea el emisor para poder modificar los bits que llegan al receptor.

La inserción es posible cuando el dispositivo objetivo tarda mucho en responder al iniciador, es ahí cuando el atacante inserta un mensaje que es una respuesta completa y si no se cumple el plan las señales enviadas se solaparan dando como resultado la corrupción de datos.

Man in the Middle

El ataque consiste en interceptar mensajes aprovechando las vulnerabilidades a través de un intermediario sin que ninguna de las dos partes se dé cuenta que la comunicación ha sido violada, con el fin de desviarla o controlarla, pasando a través de él ya que el atacante tiene conexiones con las víctimas y transmite mensajes entre ellos como si estuvieran hablando directamente, cuando lo realmente hace es inyectar nuevos.

1.2.12 Ventajas y Desventajas NFC

Las ventajas y desventajas de la tecnología NFC se presentan a continuación:

Ventajas

- Posee mayor seguridad por su rango de cobertura pequeño, garantizando ser prácticamente imposible interceptar la señal, siendo en muchos casos ideal para el intercambio de archivos sin que se pueda ver afectada la privacidad de las personas.
- Su velocidad de establecimiento y respuesta es instantánea y su uso intuitivo no necesita configuraciones previas o adicionales como emparejamiento lo que facilita la comunicación e intercambio de datos con un simple toque.
- Proporciona una excelente experiencia al usuario gracias a su sencillez, funcionalidad y versatilidad.
- Goza de gran compatibilidad con otras tecnologías y su consumo de energía es mínimo.
- Proporciona múltiples facilidades para aspectos de la vida cotidiana convirtiéndola en más sencilla por lo que la hace una tecnología única.
- NFC es muy útil en cuanto a la identificación y pagos gracias a la rapidez, comodidad y ausencia de riesgos.

Desventajas

- El intercambio de datos en esta tecnología no es masivo, no fue diseñada o pensada para la transferencia de archivos muy grandes.
- Un dispositivo en modo activo jamás podrá comunicarse con varios pasivos al mismo tiempo.
- Su rango de alcance es bastante reducido, factor que al mismo tiempo puede proporcionar varios beneficios dependiendo desde el punto de vista o necesidad.

1.2.13 Comparación con otras Tecnologías Inalámbricas

La Tabla 5-1 presenta la comparación entre las tecnologías inalámbricas más utilizadas como son NFC, Bluetooth, RFID, WIFI y Zigbee por sus características técnicas detalladas a continuación:

Tabla 5-1: Comparación de tecnologías inalámbricas por sus características técnicas

TECNOLOGÍA	*NFC*	*BLUETOOTH*	*RFID*	*WIFI*	*ZIGBEE*
Estándar	ISO/IEC 18092	IEEE 802.15.1	ISO/IEC 14443	IEEE 802.11	IEEE 802.15.4
Alcance	10 cm	10 m	3m	50-100 m	10-75 m
Frecuencia	13.56 MHz	2.4 GHz	13,56 MHz	2.4 GHz 5 GHz	868 MHz 915 MHz 2,4 GHz
Velocidad de transmisión	106, 212, 424 Kbps	720 Kbps	25 Kbps	54Mbps	20, 40, 250Kbps
Tiempo de establecimiento	< 0.1s	6s	< 0.1s	15 – 30 s	30 ms
Costo de Equipos	Mediano	Mediano	Bajo	Mediano	Ultrabajo
Seguridad	Alta	Alta si esta encriptada	Vulnerable	Vulnerable	Alta
Complejidad	Muy Sencilla	Mediana	Baja	Alta	Baja
Dispositivos interconectados	2	8	2	Indefinidos	2^16
Ancho de banda del canal	Casi 2 MHz	1MHz	2 MHz	20 MHz	5MHz
Modulación	ASK	GFSK	ASK	QPSK OFDM	DSSS
Comunicación	Dos vías	Dos vías	Una vía	Dos vías	Dos vías
Consumo de Energía	Mínimo	Alto	Mínimo	Alto	Muy bajo
Experiencia de conexión	Con un simple toque	Emparejamiento previo	Sin configuración	Configuración previa	Sin configuración

Fuente: http://goo.gl/4W1alJ
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

En referencia a la Tabla 5-1 detallada anteriormente, se puede destacar los siguientes aspectos:

NFC vs. Bluetooth

Bluetooth es una especificación industrial para Redes Inalámbricas de Área Personal que permite la transmisión de voz y datos entre dispositivos.

Las dos tecnologías son de comunicación inalámbrica en distancias cortas, NFC tiene un menor rango de alcance ya que no sobrepasa los 10cm, pero al mismo tiempo esta proximidad la hace una tecnología más segura, consume menos energía y presenta menos interferencias, muestra un mayor tiempo de establecimiento de conexión que es menor a 0.1s, por lo que es más recomendada para transmitir pequeños paquetes de datos de forma instantánea con un simple toque.

A diferencia Bluetooth alcanza los 10m, pero es menos práctica para el intercambio de datos ya que se demora más, necesita configuraciones manuales previas y emparejamiento, pero presenta una ventaja que es la comunicación con varios dispositivos al mismo tiempo.

NFC vs. RFID

RFID es un sistema que permite el almacenamiento y recuperación de datos remoto. NFC es la evolución de RFID, por lo que son tecnologías muy similares que operan a 13.56 MHz, siendo la diferencia más importante que RFID solo permite la comunicación en una única vía y NFC en dos.

La velocidad de transmisión de NFC es sumamente mayor ya que alcanza hasta 424 Kbps a diferencia de RFID con 25 Kbps, la seguridad en las etiquetas RFID es más vulnerable debido a su mayor rango de alcance y presenta más interferencias.

NFC vs. WiFi

Wifi es un sistema que permite la conexión de dispositivos de forma inalámbrica al igual que NFC, aunque la diferencia en el rango de alcance es abismal debido a que Wifi puede alcanzar los 100m y NFC tan solo 10cm, al mismo tiempo su seguridad es más vulnerable, debido a que pueden presentarse más ataques.

Wifi no cesa ya que está constantemente tratando de encontrar redes disponibles dentro de su alcance, por lo que su consumo de energía es alto y a diferencia de NFC necesita varias configuraciones adicionales, por lo que se le considera de alta complejidad en experiencia de conexión.

NFC vs. ZigBee

Zigbee es básicamente un conjunto de estándares y protocolos de comunicación inalámbrica que tiene una baja tasa de envío de información.

NFC posee una mayor velocidad de transmisión con 424 Kbps a diferencia de Zigbee con 250 Kbps.

La diferencia principal es probablemente el costo ya que ZigBee se le considera un sistema ultrabarato y consume menos cantidad de energía, pero esta tecnología no se encuentra presente en dispositivos móviles.

1.2.14 NFC en el ámbito de la Gestión Académica

En la actualidad es casi imposible concebir la vida sin la ayuda de la tecnología, ya que cada acción de una rutina diaria se puede simplificar con el apoyo de ésta.

NFC es una tecnología ampliamente utilizada para facilitar la vida de las personas, uniendo el mundo físico con el virtual, por lo que se lo utiliza en diversos campos como son la medicina, transporte, publicidad, educación, entre otros; no obstante su uso se ha enfocado en su mayoría para pagos de todo tipo.

En el ámbito educativo existen aplicaciones para el control del acceso a los estudiantes a clases, exámenes, o como un instrumento para aprendizaje de idiomas, todos ellos empleando lectores y tags NFC.

Sin embargo en lo referente a la gestión académica su utilización es escasa, cabe destacar que a pesar que en este campo existe muy poco desarrollo, sin duda alguna NFC sería muy útil en entornos educativos, abriendo un amplio abanico de posibilidades y usos en cuanto a aprendizaje, identificación y pagos.

Un ejemplo de pago en este ámbito sería el poder realizar el débito de la matrícula solo con acercar el celular, con ello disminuir los tiempos empleados y las grandes colas en el banco; otro ejemplo de uso sería el control de libros en una biblioteca.

En investigaciones realizadas se desconoce el desarrollo de sistemas de este tipo, por tal motivo nace la idea de este proyecto innovador, ya que permite transferir documentos e información académica de vital importancia, necesarios para los estudiantes al momento de realizar los diversos trámites.

1.3 MODULO OPEN HARDWARE ARDUINO

Arduino es una placa o plataforma de componentes electrónicos de código abierto, basada en hardware y software, flexible y de fácil manejo, con capacidad de apreciación o percepción del entorno a través del acoplamiento de distintos componentes externos permitiendo un enlace del mundo físico con el virtual, esta herramienta es destinada para la elaboración de prototipos, elementos o entornos interactivos.

Arduino está integrado bajo tres fundamentos que son:

1. ***Placa de hardware libre*** con un microcontrolador sencillo, reprogramable con puertos de E/S que es el encargado de procesar los datos, donde es posible conectar sensores y actuadores para las diferentes aplicaciones de Arduino.

Existen diversas placas Arduino que se diferencian por el tipo de microcontrolador integrado, funcionalidades, modelos, aplicaciones, características y número de E/S.

2. ***Software flexible, libre, gratuito y multiplataforma*** es una aplicación en forma de un entorno de desarrollo que nos posibilita programar y gestionar la memoria del microcontrolador permitiéndonos escribir, verificar y guardar el conjunto de instrucciones que definen su funcionamiento aplicativo y puede funcionar en varios sistemas operativos como Windows, Linux y Mac.

Los proyectos pueden comunicarse con el software del ordenador o ser autónomos una vez puesto en marcha luego de cargar la programación en el microcontrolador.

3. ***Lenguaje de programación libre*** es el lenguaje de máquina con que se especifican las instrucciones para que sean interpretadas y ejecutadas por los microcontroladores. El lenguaje de programación empleado es Processing/Wiring.

1.3.1 Tipos de Arduino (Hardware)

Posee varios modelos de placas con características únicas pensadas para un fin, que se pueden elegir dependiendo de la necesidad de cada usuario.

Las compañías Smart Projects, SparkFun Electronics y Gravitech son las encargadas de la manufactura y diseño de estas placas que llevan su logo, ya que son las únicas reconocidas y registradas oficialmente por Arduino.

A continuación se realizará una comparación de las placas electrónicas oficiales de Arduino por sus características técnicas:

Tabla 6-1: Comparación de los tipos de Arduino por sus características técnicas

TIPO	UNO	PRO		MEGA		NANO		PRO MINI		YÚN	LILYPAD		TRE	BT	MINI
Microcontrolador ATmega	328P	168	328	1280	2560	168	328	328 Versión 1	328 Versión 2	32U4	168V	328V	32u4 Atmel	328	328
#Terminales digitales de E/S	14	14	14	54	54	14	14	14	14	20	14	14	14	14	14
# Salidas PWM	6	6	6	15	15	6	6	6	6	7	6	6	7	6	6
# Entradas analógica	6	6	6	16	16 4 UARTs	8	8	6	6	12	6	6	6	6	8
Tensión de Funcionamiento	5V	3,3 V	5V	5V	5V	5V	5V	3.3V	5V	5V	2.7V	5.5V	5V	5V	5V
Velocidad de Reloj	16 MHz	8 MHz	16 MHz	16 MHz	16 MHz	16 MHz	16 MHz	8 MHz	16 MHz	16 MHz	8 MHz	8 MHz	16 MHz	16 MHz	16 MHz
Memoria Flash	32 KB	16 KB	32 KB	128 KB	256 KB	16 KB	32 KB	32 kB	32 kB	32 KB	16 KB	16 KB	32 KB	32 KB	32 KB
Gestor de Arranque (bootloader) usado de memoria flash	0,5 KB	2 KB	2 KB	4 KB	8 KB	2 KB	2 KB	0,5 KB	0,5 KB	4 KB	2 KB	2 KB	4 KB	2 KB	2 KB
Memoria SRAM	2 KB	1 KB	2 KB	8 KB	8 KB	1 KB	2 KB	2 KB	2 KB	2.5 KB	1 KB	1 KB	2.5 KB	2 KB	2 KB
Memoria EEPROM	1 KB	512 bytes	1 KB	4 KB	4 KB	512 bytes	1 KB	1 KB	1 KB	1 KB	512 bytes	512 bytes	1 KB	1 KB	1 KB

Fuente: http://www.arduino.cc/en/Main/Products
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

En referencia a la comparación de los tipos de Arduino especificadas en la Tabla 6-1 se concluye lo siguiente:

Debido a la gran cantidad de terminales digitales de E/S se destaca el Arduino Mega 1280 y 2560 con 54 cada uno, muy seguido del Arduino Yún que cuenta con 20 y a diferencia del resto que poseen 14.

La mayor cantidad de entradas analógicas es dada por Arduino Mega con 16, acompañado por Arduino Yún con 12, Arduino Nano con 8 y las demás con 6.

Respecto a la memoria flash que hace referencia a la capacidad de almacenamiento de los sketchs, Arduino Mega 2560 presenta superioridad con 256KB, a continuación el Mega 1280 con 128KB y los demás cuentan ya sea con 16 o 32KB.

Referente a la memoria SRAM donde se almacenan las variables principales que usa Arduino sean públicas o privadas Arduino Mega posee 8KB, seguido de Yún con 2.5 KB y el resto con 2KB.

Acerca de la memoria EEPROM donde se guarda información de manera permanente Arduino Mega posee 4KB, frente a los demás que tienen 1KB o 512 bytes.

Por consiguiente debido a las características antes mencionadas se puede optar por la elección de los tres mejores tipos de Arduino que son Arduino Mega 2560, Arduino Yún y Arduino Uno.

Cabe destacar que Arduino Mega 2560 es la versión actual de Arduino Mega 1280 y al presentar una mejora en sus características se escoge como la primera opción.

A pesar de que Arduino Uno es uno de los dispositivos más básicos, sus características son similares a los sobrantes de la lista, sumado a esto por ser el más conocido y considerado la base de todos los demás, es por ello que se convierte en la tercera opción.

Una vez establecido las tres mejores placas de Arduino por sus características detalladas anteriormente, para la posterior selección de uno de ellos, se especifica cada uno:

Arduino Uno

Arduino Uno es la más básica, la primera placa electrónica que se diseña y sale al mercado, por tal motivo se la considera la base de todas las demás y está basado en el microcontrolador ATmega328P de 8 bits, dispone de 14 entradas digitales, 6 analógicas y su voltaje de funcionamiento es 5V.

A pesar de ser una de las placas más limitadas en cuanto a la capacidad de memoria, al mismo tiempo es la más conocida y popular.

En la Figura 10-1 se aprecia la placa de desarrollo Arduino Uno, cuyas características técnicas son:

- Terminales digitales de entrada: 14 (6 pueden utilizarse como salidas PWM)
- Entradas analógicas: 6
- Voltaje de funcionamiento: 5V
- Voltaje de entrada recomendado: 7-12V
- Corriente continua para terminal I/O: 20mA
- Memoria Flash: 32KB
- Medidas Físicas: 68,6 mm de longitud, ancho=53.4 mm y peso= 25g
- Velocidad de reloj: 16 MHz

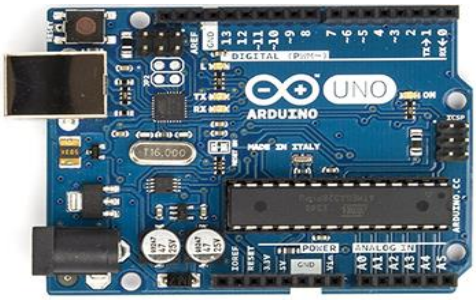


Figura 10-1: Arduino Uno
Fuente: https://www.arduino.cc/

Arduino Mega

Arduino Mega es una placa funcional, ideado con mayor número de periféricos y mayor potencia por lo que es preciso para trabajos más complejos.

Presenta el mayor número de terminales con 54 digitales, 16 analógicos, funciona a un voltaje de 5V, está basado en el microcontrolador ATmeg2560.

En la Figura 11-1 se muestra la placa de desarrollo Arduino Mega 2560, cuyas características técnicas son:

- Terminales digitales de entrada: 54 (15 pueden utilizarse como salidas PWM).
- Entradas analógicas: 16
- Voltaje de funcionamiento: 5V
- Voltaje de entrada recomendado: 7-12V
- Corriente continua para terminal I/O ATmega2560: 20mA
- Memoria Flash: 256 KB

- Velocidad de reloj: 16 MHz
- Medidas física: Longitud= 101.52 mm, ancho= 53.3 mm, peso= 37gr.

Figura 11-1: Arduino Mega 2560
Fuente: https://www.arduino.cc/

Arduino Yún

Arduino Yún es una placa que cuenta con 20 terminales digitales, 12 analógicos, funciona a 5V, presenta capacidades nativas y beneficios adicionales ya incorporados en su interior como conexión Wifi, Ethernet, ,puertos USB y almacenamiento micro-SD, gracias al chip Atheros AR9331 que posee, además puede soportar Linux y está basada en el microcontrolador ATmega32u4.

En la Figura 12-1 se muestra la placa de desarrollo Arduino Yún, cuyas características técnicas son:

- Terminales digitales de E/S: 20 (Canales PWM: 7)
- Terminales de entradas analógicas: 12
- Voltaje de funcionamiento: 5V
- Voltaje de entrada: 5V
- Corriente continua para terminal I/O: 40mA
- Memoria Flash: 32 KB
- Velocidad de reloj: 16 MHz.

Figura 12-1: Arduino Yún
Fuente: https://www.arduino.cc/

1.3.2 Entorno de desarrollo Arduino

El software es básicamente el lenguaje de comunicación con el Arduino, ya que permite gestionar la memoria del microcontrolador a través de instrucciones que éste logra interpretar.

El software Arduino es un conjunto de paquetes o elementos de software libre y código abierto, gracias a los cuales proporcionan grandes beneficios y características como bajo costo, fiabilidad, fácil comprensión, disponibilidad de acceso libre al código fuente real y a sus tecnologías implícitas, posibilitando su arreglo, modificación, ajuste o adaptación a la necesidad y completa satisfacción del usuario.

El software Arduino está licenciado bajo la versión 2 de GPL, pero sus componentes o paquetes se encuentran licenciados bajo un sinfín de concepciones distintas.

El IDE es un ambiente o entorno gráfico amigable sencillo basado en el lenguaje C++ que contiene herramientas necesarias que nos permiten la escritura y programación de lo que posteriormente deseamos compilar y ejecutar cargándoles en una variedad de placas de la familia de Arduino.

La estructura básica de la interfaz especificada en la Figura 13-1, consta de la barra de menús y shorcuts, acceso rápido que son las herramientas típicas para la gestión, el área de programación donde va el código y el área de mensajes donde muestra los estados del programa.

Figura 13-1: Entorno de desarrollo Arduino y sus partes básicas
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

1.4 ANDROID

Es una plataforma o sistema operativo móvil fundamentado en Linux, este entorno de desarrollo es sencillo, modificable, personalizable, abierto y sin costo, que fue diseñado para dispositivos inteligentes y soporta diversas tecnologías que la complementan.

En su estructura consta de diferentes capas que componen el sistema operativo determinando su funcionamiento como se muestra en la Figura 14-1.

Figura 14-1: Capas del sistema operativo Android
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

A continuación se detalla cada una de las capas que componen el sistema operativo:

Kernel de Linux

Es la parte central, corazón o núcleo, el encargado de acoger y gestionar las órdenes de los elementos y enviarlas al sistema operativo, permitiendo la comunicación del hardware con los programas, facultando de esta manera la administración de los recursos, memoria y el manejo de los procesos.

Es la parte más importante y fundamental de Android, todas estas acciones son posibles gracias a los diversos controladores para cualquier componente hardware que posee.

Librerías

Esta capa contiene las librerías nativas, donde cada una cumple tareas específicas, estas son:

- ***Surface Manager.-*** Se encarga de la gestión de la pantalla, facilitando su navegación y acceso.
- ***Media Framework.-*** Facilita el desarrollo para la reproducción de contenido multimedia.
- ***SQLite.-*** Dedicada a la concepción, almacenamiento y uso de bases de datos relacionales.
- ***SGL.-*** Suministra ambientes gráficos en dos dimensiones.
- ***Open GL.-*** Proporciona escenarios gráficos en tres dimensiones.
- ***Freetype.-*** Permite el manejo o renderización de tipos de letra, vectores o imágenes.
- ***SSL.-*** Instaura seguridad en las comunicaciones debido al cifrado que posee
- ***WebKit.-*** Provee un motor o soporte para el navegador web usado en Android.
- ***Libc.-*** Es el soporte de las demás librerías ya que contiene las funciones y cabeceras.

Android Runtime

Es realmente el entorno encargado de ejecutar las aplicaciones de Android y el responsable de correr sus programas.

Framework de Aplicaciones

Son conjuntos de herramientas que ayudan y dan soporte a la creación de las aplicaciones móviles.

Aplicaciones

Es el entorno de interacción con el usuario y el resultado final del funcionamiento de las demás capas. En Android incluyen las por defecto propias del sistema y las que la persona instale posteriormente según su requerimiento.

1.4.1 Herramientas de desarrollo Android

Android Studio trabaja conjuntamente con SDK y JDK que son herramientas indispensables, las cuales que le sirven de apoyo para el desarrollo de las aplicaciones.

Las aplicaciones de Android cuentan con el lenguaje Java y este kit para su construcción y desarrollo, SDK y JDK son un conjunto de herramientas para la creación, desarrollo, depuración y monitorización de los programas y aplicaciones que se realizan constantemente en Android.

1.4.2 IDE de desarrollo para Android

IDE o entorno de desarrollo integrado es un ambiente gráfico que cuenta con un sin número de herramientas y soporte para la creación de programas o aplicaciones, básicamente consta de un editor de código, un compilador, un depurador y un constructor de interfaz.
Android presenta dos entornos de desarrollo oficiales reconocidos y recomendados por Google Company que son: Eclipse software y Android Studio.

Tabla 7-1: Comparación de los entornos oficiales de Android por sus características

Características	***Android Studio***	***Eclipse***
Sistema de construcción	Gradle	ANT
Entorno Open Source	Si	Si
Herramientas de software para gestión y construcción de proyectos en Java	Si	No (es necesario instalar plugin auxiliar)
Construcción de variantes y diversidad de APK (Android Wear, Android TV, tablets)	Si	No
Problemas de instalación, configuración y errores en librerías.	No	Si
Refactorización y autocompletación de código	Si	No
Interfaz gráfica de diseño	Si	Si
Soporte para Google Cloud Platform	Si	No
Editor de navegación	Si	No
Firmas y claves APK (Función para firmar aplicaciones)	Si	Si
Vista previa en tiempo real (renderizado de layouts)	Si	No
Generador de assets (iconos, imágenes vectoriales, etc)	Si	No
Visualización de recursos desde editor de código y compilación desde la línea de comandos	Si	No
Reestructuración de código y soluciones rápidas	Si	No
Creación de nuevos módulos dentro de un mismo proyecto	Si	No
Plantillas para crear aplicaciones	Si	Pocas

Fuente: http://academiaandroid.com/android-studio-v1-caracteristicas-comparativa-eclipse/
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

En base al detalle de la Tabla 7-1 se aprecia que Android Studio posee superiores y mejores prestaciones en el desarrollo de aplicaciones para una diversa gama de dispositivos, a continuación se especifica éste IDE y sus componentes básicos los cuales se emplean en el desarrollo de la aplicación del sistema.

1.4.2.1 Android Studio

Android Studio es un software libre y gratuito que consta de un editor a través de un ambiente o escenario sencillo y completamente robusto, con herramientas que agilitan y facilitan la construcción de las aplicaciones en Android.

Android Studio está compuesto por diversos componentes, los cuales son:

Actividades

Es una pantalla o ventana encargada de la visualización de los elementos en la interfaz gráfica y permite la conexión entre la aplicación y el usuario final.

Las actividades constan de dos partes, gráfica y lógica, los cuales se dan en el archivo XML y .Java respectivamente.

En un proyecto la actividad debe ser establecida una por cada interfaz y cualquier elemento como botones, textos, entre otros, se definen en el fichero xml al que está asociado.

Las actividades presentan un ciclo de vida, sus estados son:

- ***Activo.-*** Es la actividad principal o en ejecución.
- ***Pausado.-*** La tarea se ejecuta y es visible pero no es la principal y pasa a un estado intermedio o de semisuspensión.
- ***Parado.-*** La tarea se finaliza por lo que se detiene y presenta un estado no visible.

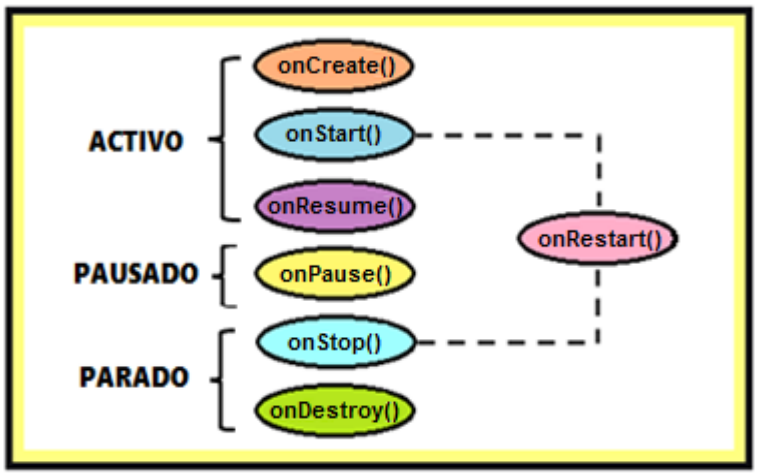


Figura 15-1: Ciclo de vida y métodos de una actividad
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

La Figura 15-1 especifica el ciclo de vida y los métodos de una actividad que intervienen en cada uno de los estados, los cuales se detallan a continuación:

Métodos

Son acciones o eventos invocados sobre una actividad que sirven para su desarrollo y definen operaciones determinadas o específicas.

Existen 7 métodos principales con los cuales se desarrolla una actividad, estos son:

- ***onCreate().-*** Método de creación, la llamada inicial o de acceso a la actividad
- ***OnRestart().-*** Permite el reinicio desde cero de una actividad luego de haberla detenido.
- ***OnStart().-*** Indica la actividad al usuario.
- ***OnResume().-*** Instaura el comienzo de la interactividad y la actividad está en primer plano.
- ***omPause().-*** Su ejecución se da si la actividad quiere dar paso a otra y pasa a un estado intermedio.
- ***onStop().-*** La actividad se detiene y pasa a segundo plano.
- ***onDestroy().***-Método de llamada final, limpieza de los recursos y desecho de la actividad.

Servicios

Son tareas o componentes que realizan cualquier tipo de acción y que se ejecutan de manera oculta y no suministra ninguna interfaz.

Intents

Son elementos que permiten el envío o petición de mensajes estableciendo la comunicación de los demás componentes y permitiendo su activación.

Vistas

Son todos los elementos básicos o detalles de la interfaz, es decir lo que se muestra en la pantalla de la aplicación como líneas, textos, botones, etc.

Proveedor de Contenido

Provee, almacena, gestiona y recupera un conjunto de datos para la aplicación facilitando su compartición.

CAPÍTULO II

2 MARCO METODOLÓGICO

2.1 Introducción

En este capítulo se da a conocer el los requerimientos del hardware, concepción de la arquitectura general, consideraciones evaluativas, análisis, selección, diseño e implementación del sistema de comunicación propuesto para el acceso a información académica.

El análisis para el proceso de selección de las mejores alternativas de la parte hardware del sistema se basa en ciertos factores importantes a tomar en cuenta a la hora de la elección e indispensables para su correcto funcionamiento, con el fin de cumplir las necesidades requeridas.

Una vez escogida la alternativa ideal que mejor se ajuste al sistema de comunicación, se procede a describir cada uno de los componentes de la parte Hardware, las respectivas conexiones, así como los softwares utilizados para el desarrollo del proyecto, dando a conocer los diagramas de flujo de los softwares a emplear.

2.2 Requerimientos del hardware del sistema

Luego del estudio realizado en el capítulo anterior se puede definir los requerimientos necesarios para el diseño y desarrollo del sistema de comunicación para el acceso a la información académica de los estudiantes basados en ocho puntos que son:

- Ser de costo moderado, razonable y de fácil instalación.
- La petición y recepción del documento hacia el usuario final debe ser de forma inalámbrica.
- La comunicación hacia la base de datos debe ser mediante cable guiado.
- Las peticiones de los usuarios deben ser una a la vez.
- El medio inalámbrico para la descarga de la información no debe necesitar ningún tipo de emparejamiento.
- La información debe ser desplegada en un smartphone.
- Se necesita un dispositivo para el control y procesamiento de datos que pueda trabajar con los dos tipos de transmisiones.
- La conexión guiada por cable debe ser Fast-Ethernet.

2.2.1 Concepción de la arquitectura general del sistema

La concepción general del sistema se aprecia en la Figura 1-2, donde se muestra el módulo de control y procesamiento de datos, el cual en conjunto con la arquitectura del software permiten enviar el requerimiento a través de la tecnología inalámbrica, dicha petición está asociada a un identificador que receptado e interpretado por el módulo de control, el mismo que realiza la consulta en el servidor y retorna la información solicitada, para su posterior impresión.

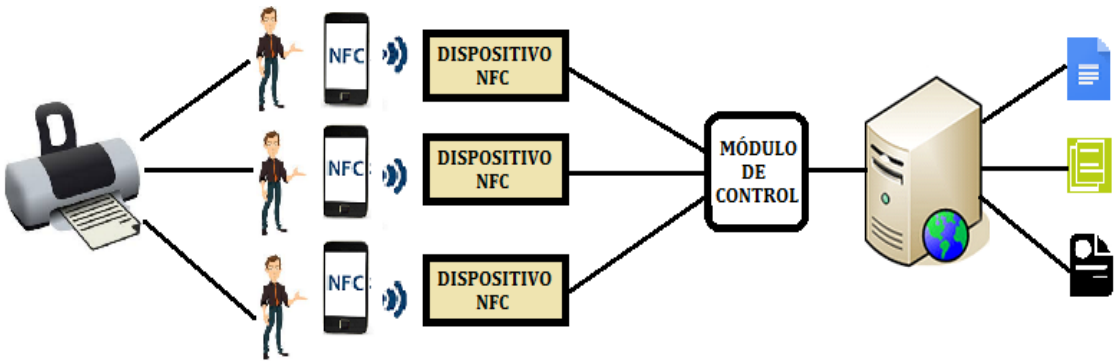


Figura 1-2: Diagrama de concepción de la arquitectura general del sistema
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

2.2.2 Diseño de la arquitectura del sistema

Una vez establecidos los requerimientos y la concepción de la arquitectura general, se presenta y detalla el diseño total de la arquitectura del sistema de comunicación. En la Figura 2-2 se muestra el diagrama de bloques, donde se especifica la interconexión con los 6 bloques que lo integran.

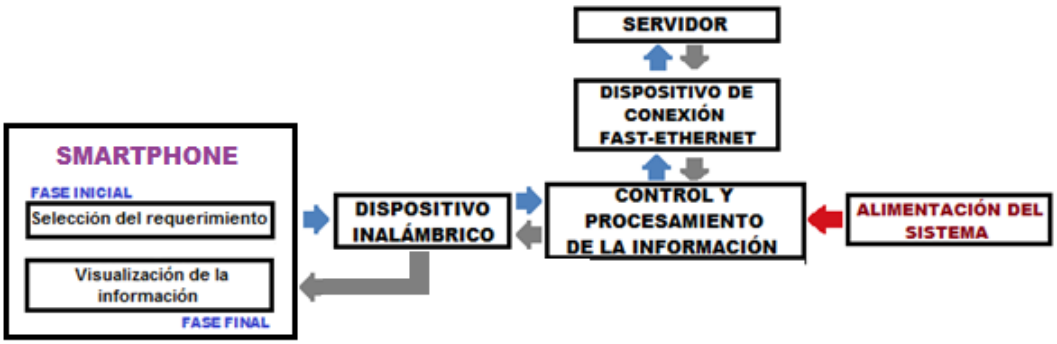


Figura 2-2: Diagrama de bloques de la arquitectura del sistema
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

El primer bloque está conformado por la etapa de envío y recepción de la petición realizada en el smartphone y enviada a través del dispositivo inalámbrico hacia el módulo de control y procesamiento de la información, el cual interpreta y realiza la consulta respectiva a través de una conexión Fast-Ethernet al servidor y posteriormente retorna la petición inicial.

El bloque de envío y recepción se conforma por la aplicación desarrollada en la plataforma Android ejecutada en un Smartphone de gama alta, a través del cual el usuario realiza las peticiones requeridas para la descarga de la información académica mediante la tecnología NFC.

El bloque de comunicación está conformado por el dispositivo inalámbrico que se encarga de enviar y retornar la petición realizada por el usuario desde la aplicación y por el dispositivo Fast-Ethernet que es el canal de transporte de los datos desde el servidor hasta el Arduino.

El bloque de procesamiento de datos es el medio de control para la transmisión y/o recepción de la información almacenada y gestionada en la base de datos.

El bloque de almacenamiento consta de la base de datos que se encuentra desarrollada en un servidor Open Source donde se guarda y consulta la información.

El bloque de alimentación es el encargado de suministrar la energía eléctrica necesaria para el funcionamiento del sistema.

La transmisión de la información se la realiza mediante una comunicación inalámbrica half-dúplex entre el teléfono y el shield NFC.

El sistema comprende un canal de comunicación compuesto en el que existe trasmisión inalámbrica y guiada por cable de red.

2.2.3 Consideraciones Evaluativas

Para determinar la tecnología inalámbrica y la placa Arduino que mejor se adapten a los requerimientos del sistema se realiza las comparaciones entre las posibles alternativas, empleando el método cualitativo por puntos.

Considerando los factores que cumplen las mejores prestaciones se ha establecido la siguiente ponderación según el criterio de evaluación de beneficios establecidos:

- Bajo valor 1
- Medio valor 2
- Máximo valor 3

En la Tabla 1-2 se establece la comparación de los factores y la asignación de su respectivo valor según el nivel de importancia, siendo uno el de mayor o igual y cero el de menor importancia.

Tabla 1-2: Valores del nivel de importancia del método cualitativo por puntos.

Comparación de factores	Valor de importancia
+ o >	1
=	1
- o <	0

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

El método cualitativo por puntos se basa en nueve pasos que son:

1. Selección de las posibles ideas.
2. Desarrollar una lista de factores relevantes o variables que condicionan las ideas.
3. Determinar los criterios del grado de importancia de cada factor con respecto al resto.
4. Realizar la tabla de las características técnicas en base a los factores del paso 2.
5. Realizar la valoración de cada factor con respecto a las ideas y detallar los totales basados en la escala relativa de ponderación y en la comparación de la tabla especificada en el paso anterior.
6. Obtener las condiciones e interpretar los criterios del paso 3 y a través de los valores especificados en la Tabla 1-2.
7. Determinar el nivel de importancia o peso.
8. Determinar de la calificación de los requerimientos.
9. Validar la idea y definir el porcentaje de aceptación de las alternativas.

2.3 Análisis y selección del Hardware del Sistema de Comunicación para el Acceso a la Información Académica.

A continuación se detalla el proceso de análisis y elección de las alternativas de Hardware a través del Método Cualitativo por puntos, con el fin de escoger la opción adecuada que satisfagan los requerimientos del sistema.

2.3.1 Alternativas de Tecnologías Inalámbricas

En el proyecto para la transmisión de datos es necesario una tecnología inalámbrica, dentro de las cuales en base a sus características y en referencia a la Tabla 5-1 especificada en el capítulo anterior, se ha considerado cuatro opciones Bluetooth, NFC, Zigbee y WiFi, por ello se realiza una comparación con el fin de determinar la tecnología adecuada a utilizar en el sistema de comunicación.

PASO 1:

- IDEA A: Tecnología Bluetooth
- IDEA B: Tecnología NFC
- IDEA C: Tecnología Zigbee
- IDEA D: Tecnología WiFi

PASO 2:

Se ha determinado 6 variables o factores que condicionan las ideas:

- F1: Seguridad
- F2: Tiempo de establecimiento
- F3: Alcance
- F4: Consumo de Energía
- F5: Costo de Equipos
- F6: Complejidad en experiencia de conexión

PASO 3:

Se tiene los siguientes criterios o políticas sobre el grado de importancia estimado de cada una de las variables a ser tomadas en cuenta a la hora de elegir la tecnología del proyecto:

- La seguridad es igual de importante que el tiempo de establecimiento y complejidad de experiencia de conexión.
- El tiempo de establecimiento es más importante que el consumo de energía y costo de equipos.
- El consumo de energía y el costo de equipos son igual de importantes.
- El alcance es menos importante que las demás variables.

PASO 4:

En la tabla 2-2 se detalla las características en base a los factores del segundo paso:

Tabla 2-2: Comparación de las cuatro alternativas de tecnologías posibles

Variables	F1	F2	F3	F4	F5	F6
Bluetooth	Media	6s	10m	Alto	60 dólares	Emparejamiento previo
NFC	Alta	<0.1s	10cm	Mínimo	40 dólares	Simple toque
Zigbee	Alta si cifrada	30ms	10-75m	Mínimo	60 dólares	Configuración previa
WiFi	Media	15–30s	50-100m	Alto	70 dólares	Configuración Previa

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

PASO 5:

Luego de realizar la valoración de cada factor con respecto a las ideas, se detalla los totales basados en la escala relativa de ponderación mencionada anteriormente y en la comparación de la Tabla 2-2:

Tabla 3-2: Peso relativo de los requerimientos de tecnología inalámbrica

Variables	F1	F2	F3	F4	F5	F6
Bluetooth	2	2	2	1	1	1
NFC	3	3	1	3	3	3
Zigbee	2	3	2	3	1	1
WiFi	2	1	3	1	1	1

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

PASO 6:

Interpretando los criterios especificados en el paso 3 se obtienen las siguientes condiciones:

- F1=F2=F6
- F2>F4, F5
- F4=F5
- F3<F1, F2, F4, F5, F6

PASO 7:

La Tabla 4-2 muestra el porcentaje importancia o peso de cada uno de los factores, para esto se basa en la Tabla 1-2 y en las condiciones interpretadas del paso anterior:

Tabla 4-2: Determinación del nivel de importancia (Peso)

FACTORES	F1	F2	F3	F4	F5	F6	Suma	Porcentaje peso
F1	-	1	1	1	1	1	5	5/19= 0.26316
F2	1	-	1	1	1	1	5	5/19= 0.26316
F3	0	0	-	0	0	0	0	0/19= 0.00000
F4	0	0	1	-	1	0	2	2/19= 0.10526
F5	0	0	1	1	-	0	2	2/19= 0.10526
F6	1	1	1	1	1	-	5	5/19= 0.26316
Total							19	1

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

PASO 8:

Tabla 5-2: Determinación de la calificación de los requerimientos

VARIABLES	F1		F2		F3		F4		F5		F6	
	VALOR	CALIF	VALOR	CALIF	VALOR	CALIF	VALOR	CALIF	VALOR	CALIF	VALOR	CALIF
Bluetooth	2	0.22222	2	0.22222	2	0.25000	1	0.12500	1	0.16667	1	0.16667
NFC	3	0.33333	3	0.33333	1	0.12500	3	0.37500	3	0.50000	3	0.50000
Zigbee	2	0.22222	3	0.33333	2	0.25000	3	0.37500	1	0.16667	1	0.16667
WiFi	2	0.22222	1	0.11111	3	0.37500	1	0.12500	1	0.16667	1	0.16667
SUMA	9		9		8		8		6		6	

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

PASO 9

Tabla 6-2: Validación de la idea y porcentaje de aceptación de tecnologías inalámbricas

VARIABLES	PESO	IDEA A: Bluetooth		IDEA B: NFC		IDEA C: Zigbee		IDEA D: WiFi	
		CALIF	VALOR	CALIF	VALOR	CALIF	VALOR	CALIF	VALOR
F1	0.26316	0.22222	0.05848	0.33333	0.08772	0.22222	0.05848	0.22222	0.05848
F2	0.26316	0.22222	0.05848	0.33333	0.08772	0.33333	0.08772	0.11111	0.02924
F3	0.00000	0.25000	0.00000	0.12500	0.00000	0.25000	0.00000	0.37500	0.00000
F4	0.10526	0.12500	0.01316	0.37500	0.03947	0.37500	0.03947	0.12500	0.01316
F5	0.10526	0.16667	0.01754	0.50000	0.05263	0.16667	0.01754	0.16667	0.01754
F6	0.26316	0.16667	0.04386	0.50000	0.13158	0.16667	0.04386	0.16667	0.04386
TOTAL			0.19152		0.39912		0.24707		0.16228
PORCENTAJE			**19,15%**		**39,91%**		**24,71%**		**16.23%**

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

RESULTADO:

Luego de evaluar todas las posibles alternativas de Tecnologías Inalámbricas a través del método cualitativo por puntos, se puede observar que la idea B es la ganadora, que pertenece a la tecnología NFC con el 39,91% de triunfo frente a sus competidoras, las cuales obtuvieron el 24.71% de Zigbee, 19.15% de Bluetooth y 16,23% de WiFi.

NFC se establece como la tecnología adecuada para el sistema de comunicación ya que proporciona ventajas como mayor seguridad, tiempo menor de establecimiento, menor consumo de energía, bajo costo de equipos y poca complejidad en experiencia de conexión para el usuario.

2.3.2 Alternativas de Plataformas Arduino

Para el módulo de control se considera las siguientes alternativas: Arduino Uno, Mega y Yún basado en el análisis de sus características ya analizadas en el capítulo anterior en la Tabla 6-1 y se realiza la comparación con el objetivo de determinar la placa adecuada a utilizar en el sistema de comunicación.

PASO 1:

- IDEA A: Arduino Uno
- IDEA B: Arduino Mega
- IDEA C: Arduino Yún

PASO 2:

Se ha determinado 4 variables o factores las que condicionan las ideas:

- F1: Costo de la placa
- F2: Velocidad de procesamiento
- F3: Número de terminales de E/S
- F4: Capacidad de Almacenamiento

PASO 3:

Se tiene los siguientes criterios o políticas sobre el grado de importancia estimada de cada una de las variables a ser tomadas en cuenta a la hora de elegir las placas:

- El número de terminales de E/S es más importante que el costo de la placa y velocidad de procesamiento.
- La capacidad de almacenamiento es más importante que las demás variables.
- El costo de la placa y la velocidad de procesamiento son igual de importantes

PASO 4:

En la tabla 7-2 se detalla las características en base a los factores del paso 2:

Tabla 7-2: Comparación de las alternativas de Arduino por sus características

Variables	F1	F2	# terminales E/S	F4
Arduino Uno	25 dólares	16MHz	14	32KB
Arduino Mega	35 dólares	16MHz	54	256KB
Arduino Yún	95 dólares	16MHz	20	32KB

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

PASO 5:

Luego de realizar la valoración de cada factor con respecto a las ideas, se detalla los totales basados en la escala relativa de ponderación mencionada anteriormente y en la comparación de la Tabla 7-2:

Tabla 8-2: Peso relativo de los requerimientos plataforma Arduino

Variables	F1	F2	F3	F4
Arduino Uno	3	3	1	1
Arduino Mega	2	3	3	3
Arduino Yún	1	3	2	1

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

PASO 6:

Interpretando los criterios del paso 3 se obtienen las siguientes condiciones:

- F3> F1, F2
- F4 > F1,F2;F3
- F1=F2

PASO 7:

La Tabla 9-2 muestra el porcentaje importancia o peso de cada uno de los factores, para esto se basa en la Tabla 1-2 y en las condiciones interpretadas del paso anterior:

Tabla 9-2: Determinación del nivel de importancia (Peso)

FACTORES	F1	F2	F3	F4	Suma	Porcentaje peso
F1	-	1	0	0	1	1/7=0.14286
F2	1	-	0	0	1	1/7=0.14286
F3	1	1	-	0	2	2/7=0.28571
F4	1	1	1	-	3	3/7=0.42857
TOTAL					7	1

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

PASO 8:

Tabla 10-2: Determinación la calificación de los requerimientos

VARIABLES	F1		F2		F3		F4	
	VALOR	CALIF	VALOR	CALIF	VALOR	CALIF	VALOR	CALIF
Arduino Uno	3	0.50000	3	0.33333	1	0.16667	1	0.20000
Arduino Mega	2	0.33333	3	0.33333	3	0.50000	3	0.60000
Arduino Yún	1	0.16667	3	0.33333	2	0.33333	1	0.20000
Suma	6		9		6		5	

PASO 9:

Tabla 11-2: Validación de la idea y porcentaje de aceptación

VARIABLES	PESO	Arduino Uno		Arduino Mega		Arduino Yún	
		CALIF	VALOR	CALIF	VALOR	CALIF	VALOR
F1	0.14286	0.50000	0.07143	0.33333	0.04762	0.16667	0.02381
F2	0.14286	0.33333	0.04762	0.33333	0.04762	0.33333	0.04762
F3	0.28571	0.16667	0.04762	0.50000	0.14286	0.33333	0.09524
F4	0.42857	0.20000	0.08571	0.60000	0.25714	0.20000	0.08571
TOTAL			0.25238		0.49524		0.25238
PORCENTAJE			**25.24%**		**49,52%**		**25,24%**

RESULTADO:

Al culminar los cálculos realizados anteriormente donde se emplea el método cualitativo por puntos, se puede definir que la mejor alternativa a elegir es la adquisición del Arduino Mega con un 49,52% de éxito en comparación con sus contrincantes que fueron el Arduino Uno y Mega con 25,24%.

Arduino Mega 2560 se establece como la placa adecuada para el sistema de comunicación gracias a sus extraordinarios beneficios adicionales como mayor número de terminales de E/S, bajo costo, mayor capacidad de almacenamiento, mayor potencia y su gran compatibilidad.

2.4 Hardware que integra el sistema desarrollado

Una vez realizado el análisis y selección de los elementos a utilizar en la parte hardware, a continuación se detalla las principales características y las conexiones respectivas necesarias para la implementación del sistema.

2.4.1 Placa Arduino Mega 2560

Arduino Mega 2560 es la placa más completa y funcional, empleada para trabajos complejos, por lo tanto es motor que controlará la transferencia de datos en el sistema de comunicación debido a que tendrá la capacidad de interpretar la petición del shield Adafruit PN532 y hacer la solicitud respectiva a la base de datos donde estará almacenada la información académica.

Adicionalmente posee conexión USB, conector de alimentación, botón de reinicio, cabecera ICSP y la posibilidad de poder ser alimentada por una fuente externa.

A continuación se presenta el gráfico de la placa de desarrollo Arduino Mega 2560:

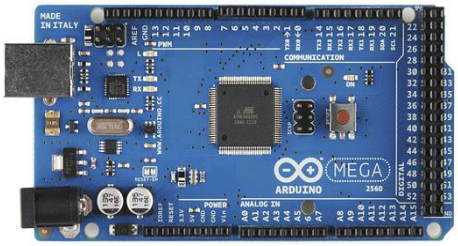


Figura 3-2: Arduino Mega 2560
Fuente: https://www.arduino.cc/

Tabla 12-2: Ficha técnica Arduino Mega 2560

Voltaje de Funcionamiento	5V
Voltaje Recomendado de Entrada	7-12V
Voltaje de Entrada (Límites)	6-20V
Pines Digitales E/S	54
Pines de Salida Digitales PWM	15
Pines de Entrada Analógicos:	16
Corriente DC por I/O Pin	20 mA
Flash Memory	256 KB
Bootloader	8KB
SRAM	8 KB
EEPROM	4 KB
Velocidad de Reloj	16 MHz
Ancho	53,3 mm
Largo	101.52 mm
Peso	37gr

Fuente: https://www.arduino.cc/en/Reference/VariableDeclaration
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

2.4.2 Shield Ethernet Arduino

Un shield es un circuito presente en una placa impresa que se conecta al Arduino con el propósito de incrementar y adicionar sus características, capacidades y funciones.

El shield Ethernet es una placa de expansión que proporciona conectividad al Arduino a través del puerto SPI, en el sistema desarrollado es el medio de transporte entre el Arduino Mega 2560 y la base de datos, permitiendo así su conexión; además permite acoplar placas adicionales encima y apilarlas como en nuestro caso el shield NFC.

A continuación se presenta el gráfico de la placa de expansión shield Ethernet Arduino:

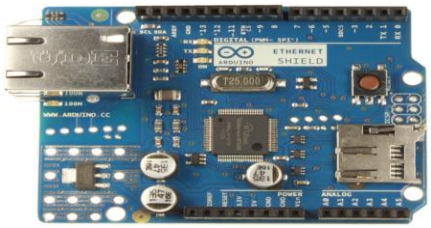


Figura 4-2: Shield Ethernet Arduino
Fuente: https://goo.gl/y815LS

Tabla 13-2: Ficha técnica Shield Ethernet Arduino

Voltaje de Funcionamiento	5V (proporcionado por Arduino)
Velocidad de conexión	10/100Mb
Velocidad de Reloj	16 MHz
Ancho	53 mm
Largo	69 mm
Peso	21gr

Fuente: https://www.arduino.cc/en/Reference/VariableDeclaration
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

Conexión con Arduino Mega 2560

El Shield Ethernet se monta sobre la placa Arduino directamente como se aprecia en la Figura 5-2, puesto que los terminales específicos que utiliza el shield vienen por defecto diseñados en el Arduino, por lo tanto sólo es cuestión de unirlos para que funcione.

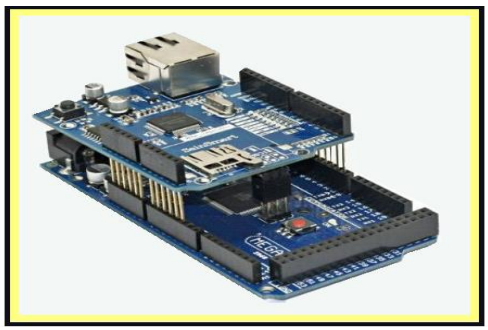


Figura 5-2: Conexión Arduino Mega y Shield Arduino Ethernet
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

2.4.3 Shield Adafruit PN532

El shield Adafruit PN532 es una placa potente de NFC y compatible con RFID capaz de leer o escribir cualquier dispositivo NFC o actuar como una tarjeta inteligente según sea el requerimiento.

A continuación se presenta el gráfico de la placa de expansión Shield NFC PN532:

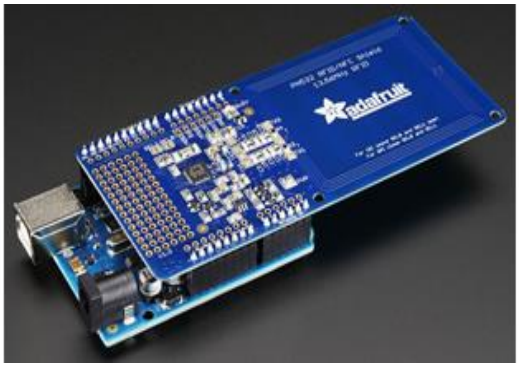


Figura 6-2: Shield Adafruit PN532
Fuente: https://www.adafruit.com/products/789

Este módulo se lo utiliza gracias a su versatilidad, compatibilidad y fácil adaptación a sistemas 5V, además que posee amplias funcionalidades que trabajan en conjunto con Arduino.

Este shield tiene dos maneras de funcionar, la primera es empleado la manera predeterminada, ocupando los terminales análogos 4 y 5, este modo se lo conoce como I2C, además cuenta con un terminal digital que realiza alertas automáticas de interrupciones, el cual por defecto es el número 2.

La segunda manera es optar por el protocolo SPI en donde se puede utilizar 4 terminales digitales cualquiera siempre y cuando se suelde 2 jumpers a la PCB superior de los módulos Arduino.

Tabla 14-2: Ficha técnica Shield Adafruit PN532

Voltaje de Funcionamiento	5V
Frecuencia	13,56 MHz
Rango de Antena	10 cm
Ancho	53,3 mm
Largo	117,7 mm
Espesor	1,1 mm
Peso	9 gr

Fuente: https://www.adafruit.com/product/789
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

Esquema de conexión con Arduino Mega 2560

Para el funcionamiento del shield Adafruit PN532 en el sistema es necesario configurarlo en modo SPI, para ello se cierra los selectores de interfaz *SEL0* y *SEL1* soldando los dos terminales de cada selector con un jumper.

El modo SPI es un protocolo síncrono de datos en serie que utilizan los periféricos para comunicarse con el dispositivo master, por lo general hay tres líneas comunes a todos los dispositivos que son:

- ***MISO.-*** Es una línea de esclavo para el envío de datos al maestro.
- ***MOSI.-*** Es la línea principal para el envío de datos a los periféricos.
- ***SCK.-*** Son impulsos de reloj que permiten la sincronización en la transmisión de datos generados por el maestro.

Además existe una línea específica para cada dispositivo llamada ***SS*** que se encarga de permitir la activación y desactivación de dispositivos específicos seleccionados por el maestro.

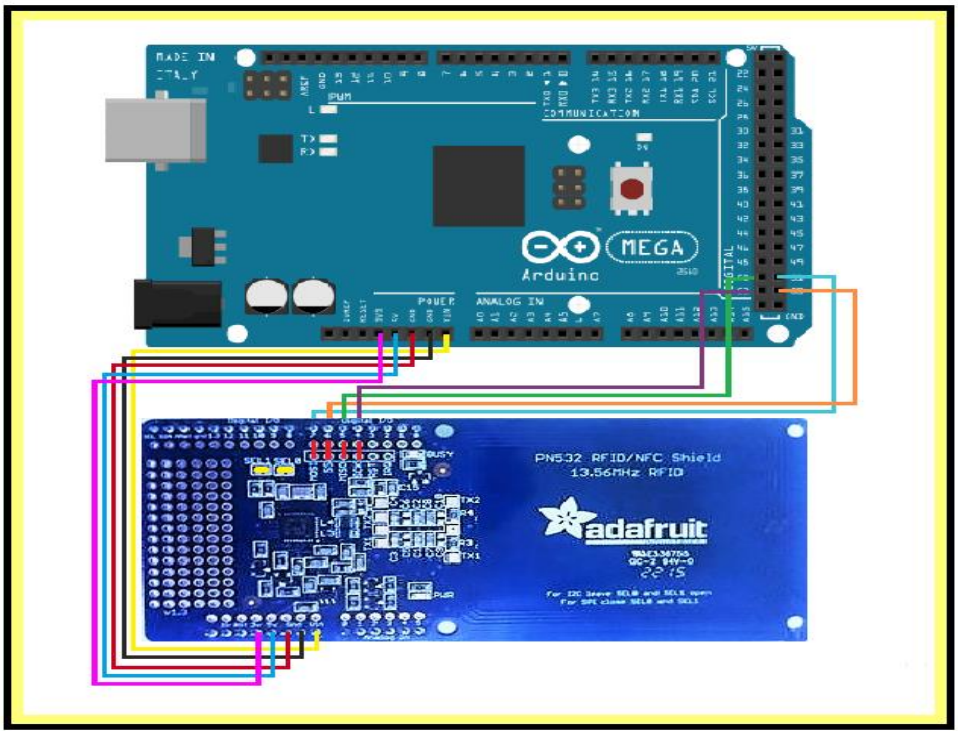


Figura 7-2: Diagrama de conexión Shield Adafruit PN532 y Arduino Mega 2560
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

La Figura 7-2 muestra la conexión del Arduino Mega 2560 y el Shield Adafruit PN532, donde para conectar dichas líneas se debe realizar las siguientes conexiones:

PASO 1:

Con el fin de utilizar los terminales digitales propios del shield NFC se realiza un puente con las líneas como se especifica a continuación:

Tabla 15-2: Conexión interna entre las líneas y terminales digitales de Shield Adafruit PN532

Línea	***Terminal Digital***
MOSI	7
SS	6
MISO	5
SCK	4

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

PASO 2:

Una vez realizado los puentes especificados en la Tabla 15-2, se procede a conectar los terminales del Shield Adafruit PN532 con los terminales correspondientes en el Arduino Mega 2560, los cuales se especifican en la siguiente tabla:

Tabla 16-2: Conexión terminales digitales de Shield Adafruit PN532 y Arduino Mega 2560

Terminal Adafruit	***Terminal Arduino***
7	51
6	53
5	50
4	52

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

2.5 Requerimientos del software del sistema

Luego del estudio realizado en el capítulo anterior se puede definir los requerimientos necesarios para el diseño y desarrollo de la etapa software del sistema de comunicación para el acceso a la información académica de los estudiantes:

1. El entorno donde la información se va a desplegar debe ser amigable e intuitivo para el usuario.
2. La información tiene que ser visualizada en una aplicación móvil.
3. La aplicación debe proporcionar información adicional como datos informativos de las autoridades y personal administrativo, croquis de las instalaciones y galería de la facultad.
4. La aplicación debe ser desarrollada en una plataforma basada en Android y en el software oficial de Google.
5. Los archivos deben almacenarse en una base de datos.
6. La base de datos debe ser desarrollada en un sistema open source.

2.6 Selección de los softwares a emplear en el desarrollo del sistema de comunicación para el Acceso a Información Académica.

En base a la conclusión del capítulo anterior de los entornos oficiales de Android, especificados en la Tabla 7-1, se determina que Android Studio es el IDE adecuado para el desarrollo de la interfaz de usuario final.

Por lo tanto la aplicación se desarrolla en Android Studio junto con sus herramientas JDK y SDK que proporcionan apoyo para su creación y desarrollo.

Arduino cuenta con su IDE oficial en el que se desarrolla la programación para establecerlo como el medio de procesamiento.

La base de datos se desarrolla en MySQL por ser un sistema open source de bajo consumo de recursos y por ocupar poco espacio en el disco, esté gestor lo incluye XAMMP que es un servidor y que permite la instalación de varias herramientas complementarias para su creación como PHP, Apache, entre otras, dentro del mismo instalador.

2.7 Desarrollo de la etapa software

Una vez especificado los programas a utilizar en la parte software, a continuación se detalla las principales características de cada uno, junto con el desarrollo de los componentes relevantes:

2.7.1 Desarrollo de la aplicación

Una aplicación se crea a partir de un sin número de archivos y carpetas por lo que para hacer la aplicación es necesario generar un nuevo proyecto.

El identificador que se envía al realizar la petición del documento está encapsulada en una trama de tres bits, el cual representa un valor numérico en binario el 0 al 14, como se especifica a continuación:

Tabla 17-2: Documentos disponibles para la descarga de información académica

DOCUMENTOS	DESCRIPCIÓN	ID
Aprobación Informe Prácticas	Solicitud para pedir la revisión y posterior aprobación del informe de prácticas pre-profesionales.	000
Asistencia regular	Solicitud para obtener certificado de asistencia regular a clases.	001
Beca Académica	Solicitud para obtención de una beca académica previo a la verificación de notas y documentación respectiva.	002
Certificación de estudio	Solicitud para obtener certificado de estar estudiando en la escuela.	003
Ciclo académico	Solicitud para obtener un certificado del ciclo académico, es decir la fecha de inicio y fin de semestre.	004
Convalidación	Solicitud para convalidación de materias, adjuntando la documentación respectiva.	005
Cupo de prácticas	Solicitud para pedir emisión de oficio para cupo de prácticas a una empresa.	006
Examen atrasado	Solicitud para rendición de exámenes atrasados, adjuntando documentación legal que justifique dicho pedido.	007
Justificación de inasistencia	Solicitud para justificación de inasistencias, adjuntando documento legal.	008
Matrícula	Solicitud para obtener certificado de matrícula y asistencia a clases.	009
Récord Académico	Solicitud para obtener el récord académico.	010
Retiro documentación	Solicitud para el retiro de la documentación para el cambio de escuela, facultad y/o institución educativa.	011
Retiro materia	Solicitud para pedir el retiro de una materia matriculada.	012
Retiro semestre	Solicitud para el retiro de todo el semestre que está cursando, adjuntando justificación.	013

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

Se puede decir que cada una de las pantallas de una aplicación son una "activity" o actividad independiente, que están conformadas por dos partes: la parte lógica (un archivo Java) y la parte gráfica (un archivo XML) como se muestra en la Figura 10-2, las cuales están desarrolladas en la App del usuario.

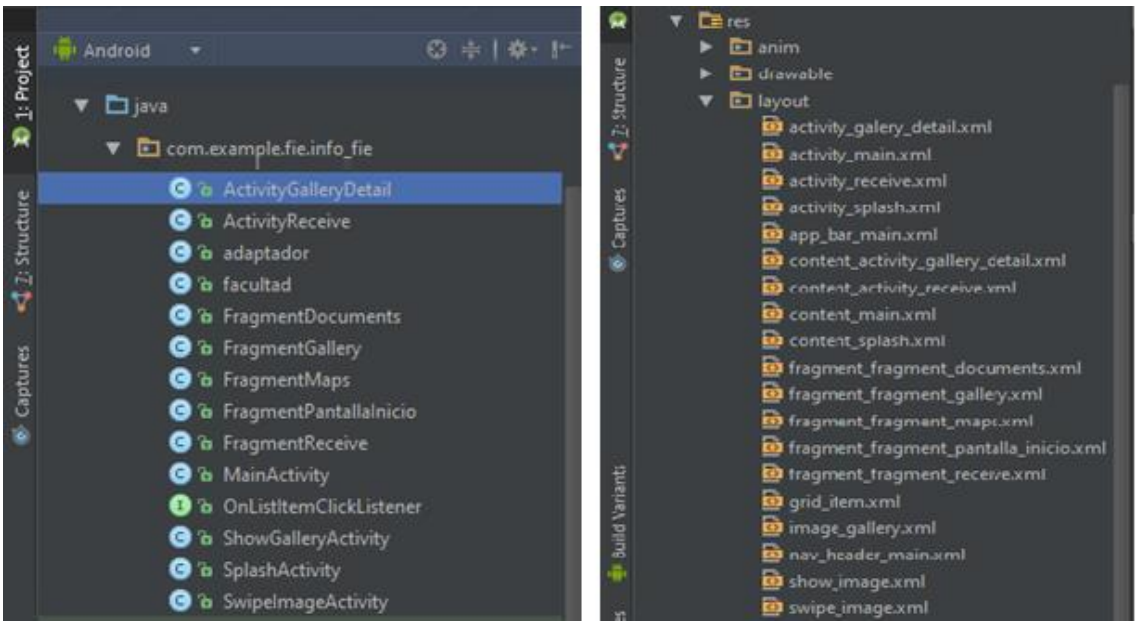


Figura 8-2: Directorio de archivos .java y .xml de App desarrollada
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

A continuación se detalla los directorios principales que conforman la App:

FragmetPantallaInicio

En la Figura 8-2 se presentan los componentes que conforman la pantalla inicial de la aplicación Info-Fie, la cual cuenta con los siguientes elementos:

- ***LinearLayout.-*** Es un contenedor de pantalla donde se organiza los componentes visuales (Button, EditText, Textiew, etc).
- ***ImageView.-*** Este control muestra imágenes en la aplicación.
- ***ScrollView.-*** Es una vista que añade un desplazamiento vertical al layout, esto permite mostrar un contenido en la pantalla proporcionando al usuario la posibilidad de desplazarse a lo largo de él.

- ***MenuButton.-*** Mediante la acción Listener despliega el menú principal de la aplicación mostrado en la Figura 9-2.

Figura 9-2: Componentes utilizados en pantalla inicio
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

FragmentDocuments

En la Figura 10-2 se presenta los componentes que conforman la pantalla del menú repositorio donde se listan los documentos que son las opciones con las que dispone la base de datos del sistema de comunicación.

- ***TextView.-*** Este control se utiliza para mostrar un determinado texto como son las distintas descripciones de los documentos.
- ***ListView.-*** Visualiza una lista deslizable verticalmente de varios elementos, que pertenecen a los documentos detallados de la base de datos.
- ***SearchView.-*** Es un widget que proporciona una interfaz para que el usuario ingrese una consulta y envíe una solicitud a un proveedor de búsqueda.

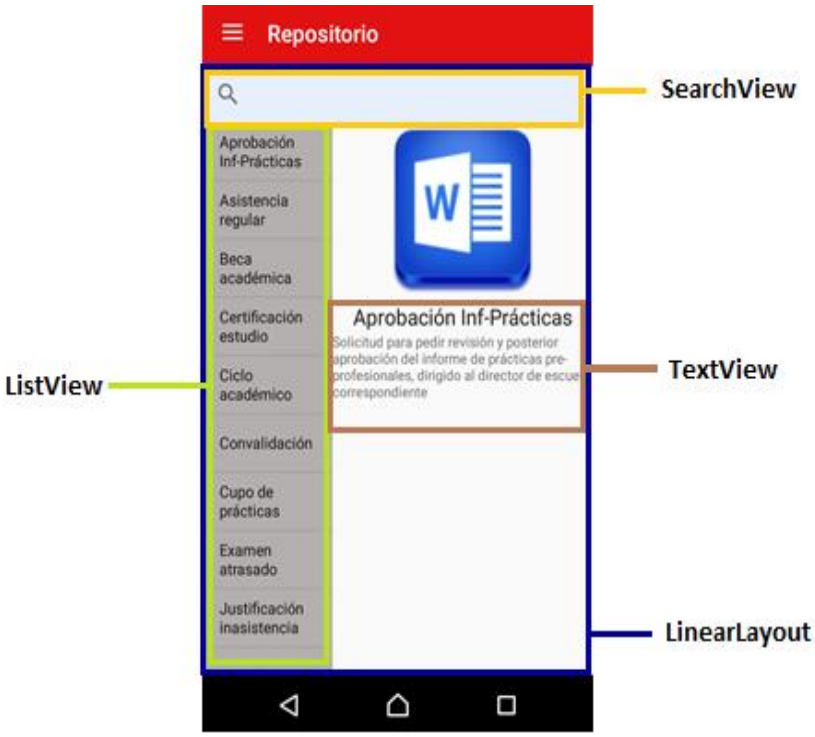


Figura 10-2: Componentes utilizados en pantalla menú repositorio
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

En la Figura 11-2 se detalla el funcionamiento básico de las pantallas que componen la aplicación. La aplicación inicia su vida útil cargando el método onCreate() en el que se cargan además de los componentes esenciales de la aplicación como librerías, variables, etc; establecen, configuran los componentes, sus respectivas funciones y llamados que son usados por la interfaz gráfica de la siguiente manera:

Una vez inicializado el método mencionado espera por el método OnItemClickListener enlazado directamente al listView que contiene la información del documento almacenado en la base de datos, donde al momento de ser seleccionado por el usuario dicho método asigna la variable respectiva del identificador a enviar, así como también a través del proceso OnListClick asigna el valor del título, descripción, e imagen a los siguientes componentes titulo.textView, descripcion.textView e imagen.imageView gracias al método setText.

Seguido se carga el método onResume() el cual contiene la variable mNfcAdapter que hace referencia al adaptador NFC del teléfono, envía el identificador sea en el caso que haya sido seleccionado algún elemento de la lista o a su vez se pone en espera para la recepción de algún documento entrante.

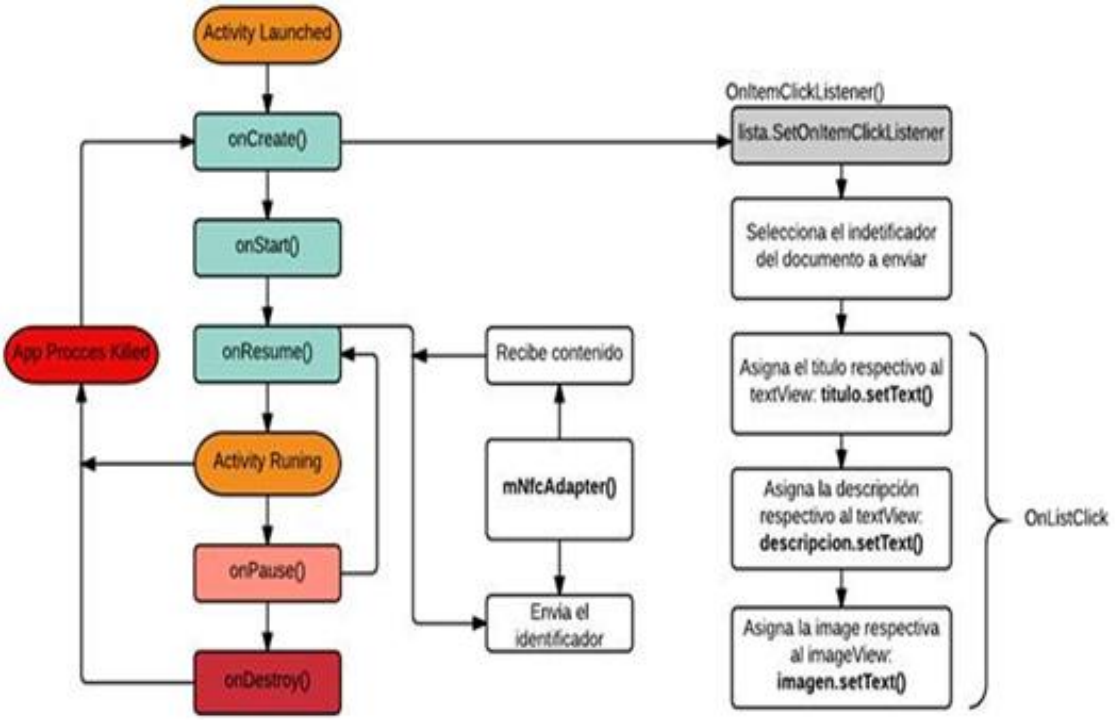


Figura 11-2: Diagrama de flujo del funcionamiento general de la aplicación
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

Cabe recalcar que el código completo desarrollado de la App Info-Fie se encuentra especificado en el anexo B.

2.7.2 Software Arduino

Es un entorno de código abierto que faculta la creación de programas para ser implementados en las placas Arduino y logrando de esta manera desarrollar aplicaciones que resuelven necesidades específicas, permite configurar la placa Arduino, estableciendo las sentencias necesarias para levantar y controlar el sistema de comunicación.

Arduino Mega 2560 es la encargada de recoger, controlar, procesar las acciones y la información en el sistema, primero interpreta los datos enviados desde el teléfono al Shield NFC, para realizar la consulta o petición al servidor mediante los métodos POST y GET en PHP, finalmente recolecta y procesa la información para ser retornada hacia el móvil.

Los requerimientos que debe cumplir son:

- Leer y validar datos NFC.
- Emitir mensajes de error.
- Realizar conexión a base de datos.
- Confirmar el requerimiento y validar la información

La Figura 12-2 muestra el diagrama de flujo donde se especifica el algoritmo de funcionamiento del módulo de control y procesamiento, basado en la placa Arduino Mega 2560. El proceso inicia con la configuración del Shield Ethernet verificando si existe direccionamiento dinámico DHCP, caso contrario asigna una IP estática, a continuación el Shield NFC espera un mensaje del iniciador que al ser confirmado lee el identificador que se envía desde la aplicación móvil.

Este identificador es enviado a la base de datos que verifica y retorna el contenido del mismo hacia el Arduino, todo eso enmarcado en un proceso interno en conjunto con PHP, si el contenido es confirmado se procede a la recepción y lectura del contenido para su posterior envío a través del Shield NFC PN532, caso contrario emite un mensaje de error.

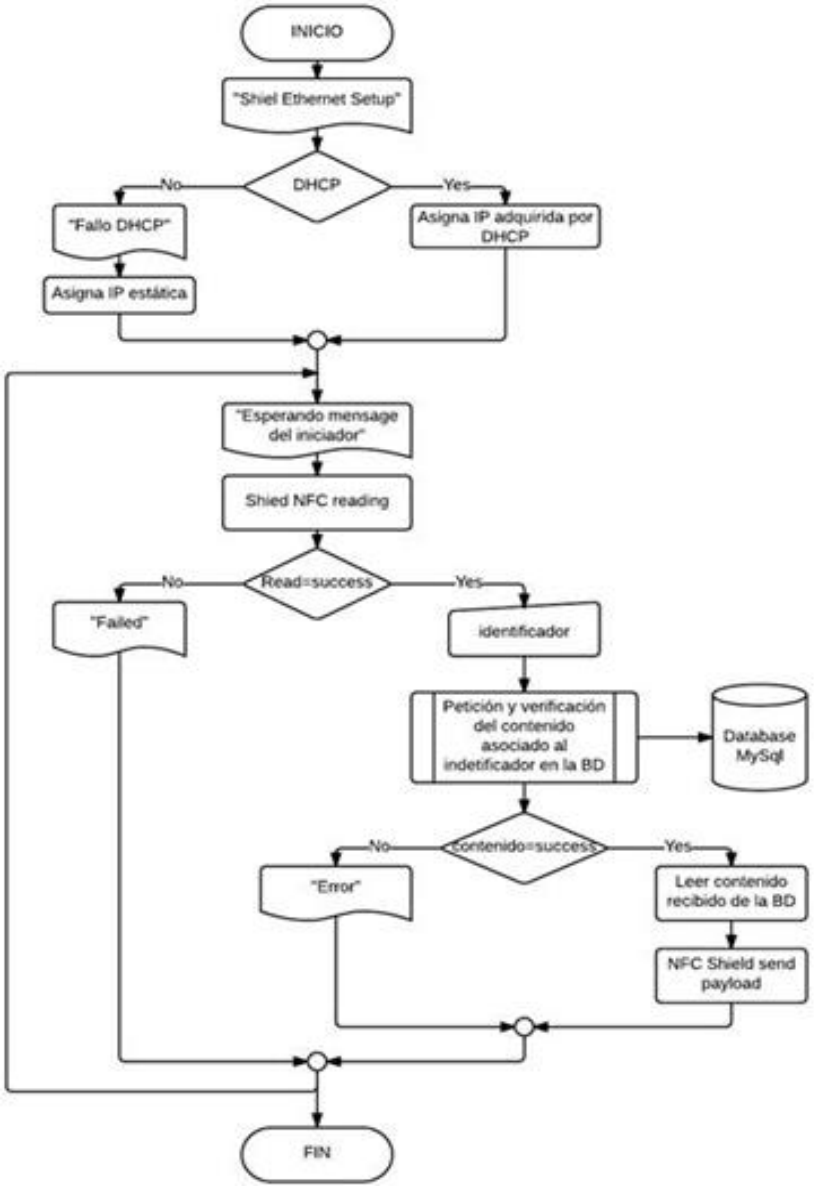


Figura 12-2: Diagrama de flujo del funcionamiento del módulo de control
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

2.7.2.1 Librerías

Para el desarrollo del sketch y el correcto funcionamiento de los dispositivos empleados es necesario la utilización de las siguientes librerías especificadas en la Tabla 18-2:

Tabla 18-2: Librerías necesarias para el funcionamiento del sistema

Dispositivo	Librería por Defecto	Librería Externa	URL externa	Descripción
Shield Ethernet	<Ethernet.h>	-	-	Proporciona conectividad Fast-Ethernet entre Arduino y cualquier red.
Adafruit PN532 NFC/RFID	-	<PN532_SPI.h>	https://github.com/don/NDEF	Permite el uso de los puertos SPI para el envío de información y control de dispositivos.
	-	<Snep.h>	https://github.com/don/NDEF	Permite la creación de la interfaz de comunicación para el Shield NFC y los dispositivos externos.
	-	<Ndefmessage.h>	https://github.com/don/NDEF	Facilita la creación y encapsulación de tramas NDEF.

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

2.7.2.2 Funciones

A continuación se detalla las principales funciones desarrolladas en el programa:

Función nfcReceive()

El Shield NFC PN532 usa esta función sin parámetros de entrada, ayuda a la recepción de tramas NDEF enviadas por el Smartphone y retornando el valor del payload o contenido recibido como una cadena de caracteres almacenada en una variable tipo String llamada payloadAsString para su posterior uso dentro del Sketch.

Internamente se vale de otras funciones como:

- nfc.read(ndefBuf, sizeof(ndefBuf))

Esta función con parámetros retorna el tamaño total del payload receptado, recibiendo una variable que es:

- **ndefBuf:** Variable tipo entero indeterminado de 8 bits de 128 posiciones, que se encarga de almacenar toda las tramas NDEF receptadas por el Shield
- **sizeof(ndefBuf):** Es una variable propia del Arduino que obtiene el tamaño de la variable ndefBuf

- NdefMessage(buffer, size)

Es una función con parámetros que retorna el contenido del payload y recibe dos variables que son:

- **buffer:** Variable tipo entero indeterminado de 8 bits de 128 posiciones, que se encarga de almacenar todo el contenido del mensaje NDEF.
- **Size:** Es una variable de tipo entero que almacena la longitud del buffer.

Función nfcSendPayload()

Es una función sin parámetros que permite el envío de envío de tramas NDEF desde el Shield NFC hacia el objetivo o teléfono móvil, las cuales contienen la información respectiva obtenida en la base de datos.

Función httpRequest()

Es una función con parámetros que permite establecer la conexión a la base de datos para realizar la respectiva consulta de la información, recibiendo un parámetro tipo int el cual es la petición inicial del usuario.

Función readPage()

Es una función sin parámetro que trabaja a la par de la función httpRequest() y permite obtener el archivo solicitado mediante un proceso de lectura por partes; además utiliza internamente dos funciones propias de Arduino que se las detalla a continuación:

- client.available()

Devuelve el número de bytes disponibles para la lectura (es decir, la cantidad de datos que ha sido escrita en el cliente por el servidor al que está conectado).

- client.read()

Lee el siguiente byte recibido del servidor al que está conectado el cliente y almacena en una variable tipo char.

Cabe resaltar que el código completo desarrollado en Arduino se encuentra en el ANEXO A.

2.7.3 Desarrollo de la base de datos en MySQL

MySQL es un software o herramienta libre visual que funciona como un servidor y sirve para gestionar o administrar bases de datos.

La base de datos creada en el proyecto permite obtener un repositorio cuya finalidad es almacenar los documentos de ayuda para los alumnos como son los formatos de solicitudes y oficios requeridos al realizar la petición de algún trámite en la facultad.

MySQL se utiliza para el diseño y es donde se crea la base de datos del sistema denominado bd_repositorio y contendrá una tabla llamada Archivos, esto podemos realizarlo gracias al gestor phpMyAdmin como se muestra en la siguiente figura:

Para el diseño de la base de datos se toma en cuenta una tabla llamada Archivos, la cual está compuesta de los siguientes campos:

- ***ID.-*** Es el identificador del documento con el cual se realiza la petición y es una variable de tipo entero.
- ***Nombre.-*** Es el nombre original del archivo, es una variable de tipo varchar con una longitud de 50.
- ***Título:*** Es el nombre específico que se le da al archivo, es de tipo varchar de longitud 50.
- ***Descripción.-*** Es la explicación más detallada del documento, es de tipo varchar de longitud 100.
- ***Contenido.-*** Es la variable donde está almacenado el archivo, es de tipo MediumBlob y puede con una capacidad de hasta 16MB.
- ***Tipo***: Es una variable de tipo varchar de longitud 50, contiene el MIME Type de cada Archivo, que es un estándar que se utiliza para enviar contenido a través de la red.

Tabla 19-2: Ejemplo de tabla Archivos de base de datos

ARCHIVOS					
ID	***Nombre***	***Título***	***Descripción***	***Contenido***	***Tipo***
1	Doc1.doc	Oficio_Cambio_Carrera	Solicitud para traslado de carrera	[BLOB]-13KB	application/msword
2	Doc2.pdf	Oficio_Convalidación	Solicitud para convalidación de materia	[BLOB]-12KB	application/pdf
…	…	…	...	…	…
N	DocN	Oficio_N	Solicitud N	[BLOB]- N	application/N

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

CAPÍTULO III

3 MARCO, DISCUSIÓN Y ANÁLISIS DE RESULTADOS

3.1 Introducción

En este capítulo se determina la verificación del funcionamiento, planificación de pruebas, evaluación, recolección, análisis de datos obtenidos para determinar los resultados del presente trabajo, se constata cada etapa mediante imágenes logrando el establecimiento del sistema, así como su funcionamiento completo, lo cual permite cumplir con el objetivo principal que es poseer un sistema de comunicación con NFC para el acceso a la información académica de los estudiantes de la FIE-ESPOCH.

Para poder establecer que el sistema de comunicación realizado es una excelente alternativa para el acceso a la información académica y con ello la agilización de los procesos, se procede a realizar una comparación bajo dos puntos; el primero es el método tradicional o actual de petición de solicitudes en la secretaría de cada escuela y el segundo la medición en tiempo real de una descarga en el sistema de comunicación.

3.2 Verificación del funcionamiento del sistema de comunicación

La Figura 1-3 determina el diagrama de bloques del funcionamiento total del sistema de comunicación desarrollado.

Al suministrar la alimentación al sistema permite iniciar la unidad de control programada sobre Arduino Mega 2560 y conectado a dos shield: El Ethernet el cual se enlaza a la base de datos creada en My SQL para la visualización o consulta de archivos y al Adafruit PN 532 que es el encargado de retornar el documento hacia la aplicación desarrollada en Android Studio e instalada en un móvil NFC.

Cabe destacar que el sistema de comunicación es colocado en una caja segura para evitar daños ambientales por el clima, contacto o mala manipulación de usuarios, daños o averías en circuitos o cualquier situación que pueda provocar fallas.

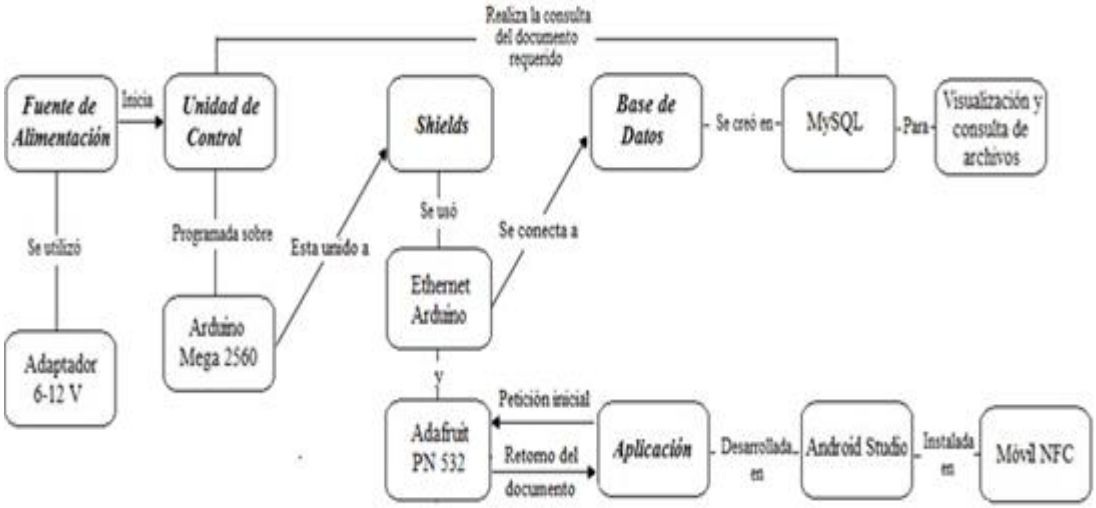


Figura 1-3: Diagrama de bloques de funcionamiento del sistema.
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

3.2.1 Planificación de Pruebas

La planificación de pruebas se obtiene bajo dos fases de evaluación, la primera trata de la verificación del bloque de comunicación donde se medirán los siguientes parámetros: Obtención de la dirección IP, medición del rango de operación y descarga de la información.

La segunda fase evalúa la integridad de los datos, cuyo propósito es determinar si la variación en el rango de operación adecuado del sistema altera o no el tiempo de descarga.

3.2.1.1 Evaluación del bloque de comunicación

La evaluación del bloque de comunicación funciona bajo tres parámetros que son:

Obtención de dirección IP del sistema

Al iniciar el sistema se realiza la obtención de la dirección IP mediante el protocolo dinámico DHCP, que se puede monitorear a través en la conexión del puerto USB.

Luego de las pruebas realizadas se puede visualizar y comprobar que la conexión se ha establecido (Figura 2-3) y que cumple con los requerimientos del diseño elaborado.

Figura 2-3: Obtención de dirección IP
Realizado por: Lara, Daniela; Vallejo, Javier, 2016

Prueba de determinación del rango de operación

En esta etapa se realiza la descarga de los 10 documentos al azar almacenados en la base de datos, como se aprecia en la Tabla 1-3, donde se obtiene el número de palabras, caracteres y el tiempo de descarga.

El proceso inicia enviando la petición desde la aplicación en el teléfono móvil (Figura 3-3), la cual contiene el ID correspondiente al documento solicitado, que es receptado y posteriormente hecha la consulta de la información asociada a la misma en la base de datos.

Para ello se ingresa en la opción repositorio donde despliega las opciones de todas las solicitudes junto con la descripción de cada una.

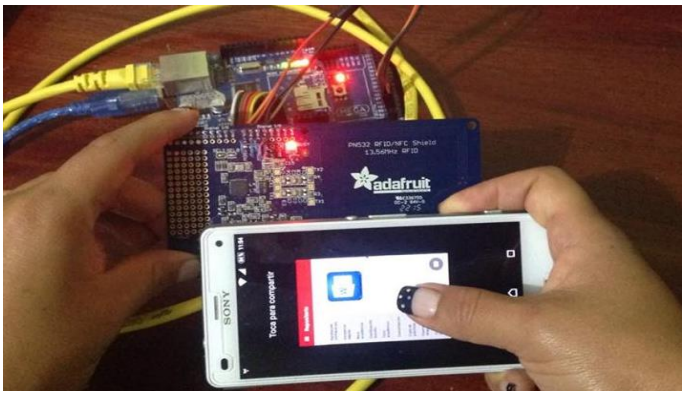


Figura 3-3: Petición al sistema del documento
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

Tabla 1-3: Verificación de la distancia de funcionamiento

Distancia	***Palabras***	***Caracteres***	***Tiempo***
10 cm	103	790	No descarga
9 cm	100	785	No descarga
8 cm	92	763	No descarga
7 cm	105	792	No descarga
6 cm	90	740	0,29 segundos
5 cm	102	780	0.31 segundos
4 cm	104	725	0.31 segundos
3 cm	98	736	0.32 segundos
2 cm	100	785	0,41 segundos
1 cm	101	710	0,43 segundos

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

En base a los datos obtenidos en la Tabla 1-3 se determina que en el rango de funcionamiento máximo es de 6 cm debido a que la conexión es satisfactoria y la descarga exitosa en un tiempo promedio de 0,346 segundos.

Se puede destacar que en rangos de 7 a 10 cm no se establece conexión, a pesar de que las especificaciones técnicas detallan que el rango máximo de funcionamiento es de 10cm.

Prueba de descarga de la información

Esta etapa se observa la transferencia de los datos y posterior recepción, se puede constatar de dos maneras:

La primera opción es mediante el monitoreo USB como se muestra en la Figura 4-3, donde el cuadro de color rojo es el ID que llega desde la petición del teléfono, el de color celeste es el mensaje que indica la conexión satisfactoria con la base de datos y el amarillo es la información codificada y en proceso de transferencia.

La segunda manera es visualizar directamente en el teléfono visualizando el documento requerido como se muestra en la Figura 5-3.

Figura 4-3: Verificación del envío y recepción de datos.
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

Figura 5-3: Verificación de datos recibidos en el móvil.
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

3.2.1.2 Evaluación del rango de operación Vs. El tiempo de descarga

Esta evaluación consta en medir la descarga de un mismo documento en el rango operativo del sistema en base a la conclusión de la Tabla 1-3 y determinar si el tiempo de descarga varia, como se puede apreciar en la siguiente tabla:

Tabla 2-3: Evaluación rango de operación Vs. tiempo de descarga

Rango de operación	*Caracteres*	*Tiempo de descarga*
6 cm	104	0,31 segundos
5 cm	104	0,31 segundos
4 cm	104	0,31 segundos
3 cm	104	0,31 segundos
2 cm	104	0,31 segundos
1 cm	104	0,31 segundos

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

En base a los datos obtenidos en la Tabla 2-3 se determina que la descarga de un mismo documento en los distintos rangos no se ve afectado por el tiempo de descarga.

3.3 Requerimientos del sistema de comunicación desarrollado

Los requerimientos son ciertos pasos o restricciones con las cuales el sistema funciona apropiadamente y deben ser impartidos al usuario para la correcta descarga de la información, éstos son:

- El usuario debe poseer un teléfono inteligente que cuente con la tecnología NFC.
- El usuario deberá instalar previamente la aplicación sin ningún costo.
- El dispositivo debe contar con la versión de Android 4.0 hasta 4.4

3.4 Recolección y Análisis de Datos

A continuación se realizará la comparación del método actual con el sistema de comunicación desarrollado para establecer las respectivas diferencias y mejoras.

3.4.1 Método Actual

Para la recolección de los datos se ha realizado una medición real del tiempo en los trámites de solicitudes en la secretaría de la escuela, donde el proceso tradicional se desarrolla de la siguiente manera:

- PASO 1: Ingresar a la secretaría de cada escuela en su ubicación correspondiente.
- PASO 2: Esperar el turno de cada persona, es decir la disponibilidad.
- PASO 3: Acercarse y pedir información junto con el formato de la solicitud.
- PASO 4: Esperar a que se realice la búsqueda de los archivos en carpetas o en la computadora.
- PASO 5: Obtener la información requerida e ir a redactar el oficio para posteriormente regresar y realizar el respectivo trámite.

Estos cinco pasos detallados anteriormente es el proceso actual que se realiza cada día, en una mañana se atiende un promedio de 15 personas, por lo que se ha tomado los datos o muestras de 10 (Tabla 3-3), con el fin de establecer el tiempo promedio de demora.

Tabla 3-3: Tiempo estimado de método tradicional.

Persona	Documento (Solicitud)	Tiempo pedido de información	Tiempo pedido de información
1	Convalidación de Materias	4 min y 5 seg	245 seg
2	Matrícula	3 min y 57 seg	237 seg
3	Aprobación Informe de Prácticas	5 min y 10 seg	310 seg
4	Récord Académico	2 min y 45 seg	165 seg
5	Retiro de Materia	3 min y 50 seg	230 seg
6	Matrícula	4 min y 25 seg	265 seg
7	Récord Académico	3 min y 40 seg	220 seg
8	Justificación de inasistencia	4 min y 58 seg	298 seg
9	Matrícula	2 min y 39 seg	159 seg
10	Récord Académico	1 min y 21 seg	81 seg

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

Gráfico 1-3: Tiempo empleado por 10 personas en el sistema tradicional.
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

RESULTADO:

Al aplicar el método estadístico media aritmética de las muestras obtenidas se aprecia que el tiempo estimado en demora de este tipo de trámite por persona es de 3 minutos y 41 segundos, equivalente a 221 segundos.

3.4.2 Sistema de Comunicación con NFC

En la recolección de los datos se ha realizado una medición real del tiempo de la descarga de las solicitudes e información en el sistema de comunicación, los mismos que están detallados en la Tabla 4-3.

Cabe recalcar que es el tiempo total empleado desde la petición inicial realizada por el usuario a través de la aplicación instalada en el móvil, hasta el retorno de la información para los trámites necesarios.

Tabla 4-3: Tiempo empleado de descarga de sistema de comunicación desarrollado.

Documento (Solicitud)	Peso	Tiempo tecnología NFC	Tiempo sistema de comunicación con NFC	Tiempo sistema de comunicación
Aprobación Informe de prácticas pre-profesionales	13KB	0.031 seg	1 min y 0.031seg	60.031 seg
Asistencia Regular	12KB	0.029 seg	1 min y 0.029seg	60.029 seg
Beca Académica	12KB	0.029 seg	1 min y 0.029seg	60.029 seg
Ciclo Académico	13KB	0.031 seg	1 min y 0.031seg	60.031 seg
Convalidación de Materias	13KB	0.031 seg	1 min y 0.031seg	60.031 seg
Estudio Regular	12KB	0.029 seg	1 min y 0.029seg	60.029 seg
Exámenes Atrasados	12KB	0.029 seg	1 min y 0.029seg	60.029 seg
Justificación de Inasistencia	12KB	0.029 seg	1 min y 0.029seg	60.029 seg
Matrícula	13KB	0.031 seg	1 min y 0.031seg	60.031 seg
Cupo de prácticas pre-profesionales	13KB	0.031 seg	1 min y 0.031seg	60.031 seg
Récord Académico	12KB	0.029 seg	1 min y 0.029seg	60.029 seg
Retiro de documentación para cambio	12KB	0.029 seg	1 min y 0.029seg	60.029 seg
Retiro de Materia	12KB	0.029 seg	1 min y 0.029seg	60.029 seg
Retiro total de semestre	13KB	0.031 seg	1 min y 0.031seg	60.031 seg

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

RESULTADO:

Al aplicar el método estadístico media aritmética de los valores obtenidos en el sistema de comunicación, se aprecia que el tiempo estimado para la descarga es de 1 minuto y 0.03 segundos, lo que es equivalente a 60.03 segundos.

3.4.3 Método Actual Vs. Sistema de Comunicación con NFC

En los cálculos realizados anteriormente se puede ver que el tiempo aproximado por persona empleando el método actual es de 221 segundos, mientras que en el sistema de comunicación con NFC ocupa 60.03 segundos, para poder establecer una relación y visualizar la gran diferencia que existe se compara estos tiempos empleados en cinco personas:

Tabla 5-3: Comparación de tiempos empleados entre sistemas.

Cantidad de Personas	Tiempo del método actual (segundos)	Tiempo sistema de comunicación (segundos)
Una	221	60.03
Dos	442	120.060
Tres	663	180.09
Cuatro	884	240.120
Cinco	1105	300.15

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

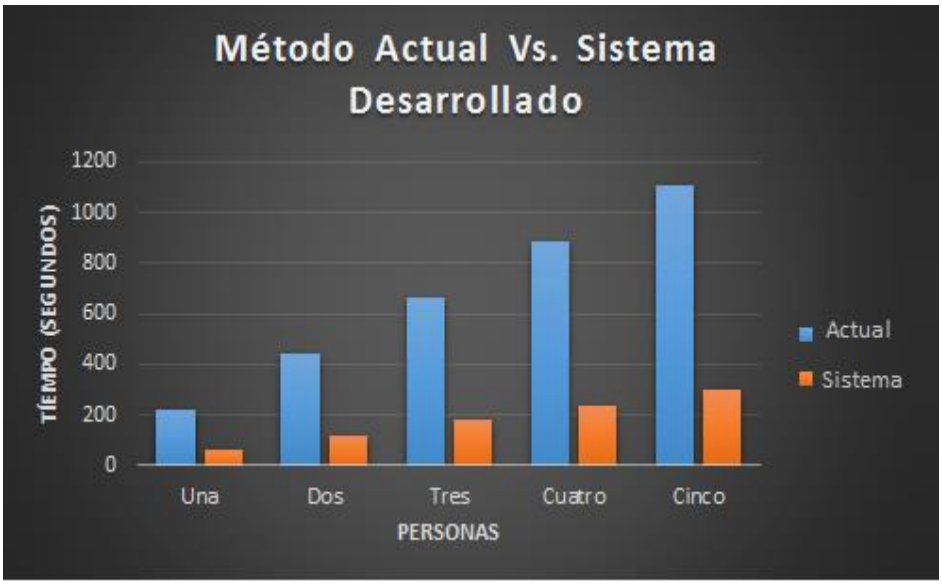


Gráfico 2-3: Relación método actual vs. Sistema de comunicación con NFC
Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

Como se aprecia claramente en el Gráfico 2-3, el sistema de comunicación posee una mejor relación respecto al tiempo empleado en comparación con el método actual, por lo que es una excelente alternativa ya que sin lugar a duda mejora y agilita los procesos.

3.5 Análisis de Resultados

Para calcular el nivel de mejora, se procede a hallar el valor denominado Δs que es la diferencia entre el método tradicional y el sistema realizado, con el propósito de establecer el porcentaje de mejora del sistema implementado.

$$\Delta s = (221 - 60.03)s = 160.970\ segundos$$

$$\Delta s = 72.84\%$$

Por lo tanto se puede observar una diferencia abismal de aproximadamente 2 minutos 41 segundos, cumpliendo con un 72,84% de eficiencia y nivel de mejora frente al método actual.

3.6 Análisis de Corriente

En base a las especificaciones y características técnicas de cada dispositivo que conforma el sistema de comunicación abarcadas en los capítulos anteriores, se puede determinar que es un sistema de bajo consumo eléctrico ya que ocupa 37mA, especificados a continuación:

Tabla 6-3: Valor de corriente de los dispositivos del sistema de comunicación

Dispositivo	Valor de corriente
Arduino Mega 2560	20mA
Shield Ethernet	15Ma
Shield NFC PN352	2Ma

3.7 Análisis Económico

A continuación se detalla los costos de cada uno de los componentes empleados en el desarrollo del sistema de comunicación, con los elementos adquiridos en Estados Unidos.

Tabla 7-3: Listado de componentes y costos del sistema de comunicación.

Cantidad	Descripción	Subtotal
1	Arduino MEGA 2560	30$
1	Módulo NFC	60$
1	Shield Arduino Ethernet	20$
1	Varios (Caja, logos, transportes)	50$
1	Smartphone con NFC	450
Descripción		**Subtotal**
Arduino software		0$
Android Studio		0$
PHP		0$
XAMMP		0$
TOTAL		**610$**

Realizado por: Lara, Daniela; Vallejo, Javier, 2016.

En la Tabla 5-3 se presenta el listado de componentes y costo del sistema de comunicación, el cual es de 160 dólares americanos sin tomar en cuenta el valor del smartphone, siendo de bajo costo y cumpliendo así con el requerimiento del diseño del sistema.

CONCLUSIONES

- Se desarrolló un sistema de comunicación basado en la tecnología inalámbrica NFC, permitiendo a los estudiantes obtener documentos necesarios a la hora de realizar diversos trámites e información adicional de la facultad. Trabaja a un rango de funcionamiento máximo de 6cm y con un voltaje de entrada de 5 Voltios por conexión USB o 7 V por alimentación externa.

- El método cualitativo por puntos permitió realizar la comparación, el análisis de las distintas tecnologías inalámbricas y determinar que NFC posee las características adecuadas que satisfacen los requerimientos del diseño del sistema con un 39,91% de mejoras frente a sus competidoras, proporcionando ventajas como ahorro de energía, tiempo de establecimiento bajo lo que permite disminuir el tiempo empleado y experiencia simple en conexión para el usuario debido a que no se necesitan configuraciones previas y el típico emparejamiento.

- Al realizar el estudio y la comparación por el método cualitativo por puntos de los diferentes modelos de placas reducidas Arduino, se determinó que Arduino Mega 2560 es la más favorable para ser implementada como unidad de control y procesamiento con un 49,52% a favor, debido a la mayor capacidad de almacenamiento y mayor número de terminales de E/S que presenta.

- El desarrollo de la aplicación para teléfonos móviles se realizó sobre la plataforma Android Studio con un tamaño de 20 MB, es gratuita, libre para su descarga, permitiendo la comunicación entre el usuario y el sistema,

- En las pruebas realizadas se determinó que la descarga de información académica del sistema electrónico de comunicación desarrollado se realiza en un tiempo promedio de 60 segundos por documento, presentando una diferencia de aproximadamente 161 segundos con respecto al método actual y proporcionando un nivel de mejora de 72,84% de eficiencia.

- El diseño del sistema se desarrolló en una configuración tipo estrella y de forma escalable, lo que permite implementar nuevos módulos. Su uso contribuye en la mejora del tiempo de atención al estudiante.

RECOMENDACIONES

- El presente trabajo puede ser base para la elaboración e implementación de sistemas en el ámbito de la gestión académica como para el pago de matrícula por arrastre, monitoreo de personal, control de seguridad en puertas, entre otros.

- En el posterior desarrollo implementar en la aplicación algún sistema de autenticación que permita al usuario registrarse y tener acceso a un mayor número de recursos e información.

- Incluir en el desarrollo futuro otro tipo de información como pensum académico, mallas curriculares, horarios, etc, de beneficio para el estudiante.

- Continuar con el desarrollo y mejora del prototipo para una posible comercialización en pos de presentar una solución en un ambiente real y acorde a las necesidades del medio.

- Instalar un servicio de impresión al sistema para dinamizar aún más los tiempos de respuesta.

BIBLIOGRAFÍA

1. **ALBECA GÓMEZ, Edwin Leodan.** *Estudio de la Tecnología Inalámbrica NFC (Near Field Communication) y sus Aplicaciones en el Ámbito de las Telecomunicaciones* [En línea] (tesis pregrado). Escuela Politécnica Nacional, Facultad de Ingeniería Eléctrica y Electrónica, Quito, Ecuador. 2013. pp. 37-48. [Consulta: 2016-03-02]. Disponible en: http://bibdigital.epn.edu.ec/handle/15000/6440

2. **ALONSO RODRIGUÉZ, José María.** *Diseño e Implementación de un lector PC/SC inalámbrico para tarjeta inteligente basado en plataformas móviles NFC* [En línea] (tesis pregrado). Universidad de Cantabria, Escuela Técnica Superior de Ingenieros Industriales y de Telecomunicación, Cantabria, España. 2013. pp.6-7. [Consulta: 2016-03-23]. Disponible en: http://repositorio.unican.es/xmlui/bitstream/handle/10902/1907/352901.pdf?sequence=6

3. **AUCANCELA GUAGCHA, Edwin Iván.** *Guía Metodológica para la implementación de la tecnología NFC para pagos en Línea en Instituciones Financieras* [En línea] (tesis pregrado). Escuela Superior Politécnica de Chimborazo, Facultad de Informática y Electrónica, Escuela de Ingeniería en Sistemas, Riobamba, Ecuador. 2015. pp. 45-48. [Consulta: 2016-03-25]. Disponible en: http://dspace.espoch.edu.ec/bitstream/123456789/4386/1/18T00586.docx.pdf

4. **CACUANGO GUACHALA, Darwin Segundo.** *Diseño e Implementación de un Sistema de Control de Acceso Multinivel en base de Receptores Near Field Communication (NFC)* [En línea] (tesis pregrado). Universidad Politécnica Salesiana, Escuela de Ingeniería Electrónica, Quito, Ecuador. 2015. pp. 25-29. [Consulta: 2016-03-07]. Disponible en: http://dspace.ups.edu.ec/bitstream/123456789/10160/1/UPS%20-%20ST001789.pdf

5. **CANTELLÁN, Anaya A, & LÓPEZ MARTÍNEZ, I.** *La tecnología NFC en teléfonos celulares, sus retos y aplicaciones.* [En línea] (Artículo posgrado). Instituto Tecnológico de Orizaba, Orizaba, México. 2014. pp.100-102. [Consulta: 2016-01-22]. Disponible en: http://www.rcs.cic.ipn.mx/2014_77/La%20tecnologia%20NFC%20en%20telefonos%20celulares_%20sus%20retos%20y%20aplicaciones.pdf

6. **CHAVARRÍA CHAVARRÍA, Daniel Antonio.** *Tecnología de Comunicación de Campo Cercano (NFC) y sus Aplicaciones* [En línea] (tesis pregrado). Universidad de Costa Rica, Facultad de Ingeniería, Escuela de Ingeniería Eléctrica, Ciudad Universitaria Rodrigo Facio, Chile. 2011. pp. 34-37. [Consulta: 2016-02-11]. Disponible en: http://eie.ucr.ac.cr/uploads/file/proybach/pb2011/pb2011_012.pdf

7. **EVANS, B.** *Arduino Notebook a beginner´s reference* [en línea]. San Francisco California-USA: Creative Commons, 2007. [Consulta: 20 marzo 2016]. Disponible en: http://playground.arduino.cc/uploads/Documentation/Arduino_programming_notebook.pdf

8. **INTECO.** *La Tecnología NFC: Aplicaciones y Gestión de Seguridad* [pdf].2004. [Consulta: 29 marzo 2016]. Disponible en: http://www.egov.ufsc.br/portal/sites/default/files/cdn_nfc_final.pdf

9. **INVARATO, R.** *Android 100%* [en línea]. Madrid, España. Creative Commons. 2014. [Consulta: 28 de Abril 2016]. Disponible en: http://jarroba.com/libro-android-100-gratis/

10. **JUÁREZ GUTIERREZ, Beatriz.** *Desarrollo de una aplicación NFC en un entorno universitario con autenticación basada en el Elemento Seguro* [pdf] (tesis pregrado). Universidad Carlos III de Madrid, Escuela de Ingeniería Técnica de Telecomunicación Telemática, Leganés, España. 2011. pp. 29-31. [Consulta: 2016-03-02]. Disponible en: http://e-archivo.uc3m.es/bitstream/handle/10016/11932/PFC_BEATRIZ_JUAREZ_GUTIERREZ.pdf?sequence=1

11. **NFC-FORUM**. *NFC Data Exchange Format (NDEF)* [en línea]. Wakefield Boston-USA: NFC Forum, Inc. 2006. [Consulta: 25 de marzo 2016]. Disponible en: http://www.eet-china.com/ARTICLES/2006AUG/PDF/NFCForum-TS-NDEF.pdf

12. **PADILLA CONTRERAS, Jorge Esteban, & LÓPEZ IÑIGUEZ, Wilber Adrián**. *Near Field Communication-Teoría y Aplicaciones* [En línea] (tesis pregrado). Universidad del Azuay, Facultad de ciencias de la Administración, Escuela de Ingeniería de Sistemas y Telemática, Cuenca, Ecuador. 2014. pp. 38-41. [Consulta: 2016-03-28]. Disponible en: http://dspace.uazuay.edu.ec/bitstream/datos/3586/1/10270.pdf

13. **PENALVA, Javier.** *NFC: qué es y para qué sirve* [blog]. *Murcia-España: XATACA, 2011.* [Consulta: 14 de enero 2016]. Disponible en: http://www.xataka.com/moviles/nfc-que-es-y-para-que-sirve

14. **SÁNCHEZ MANZANOS, Arturo.** *Aplicación de la tecnología de comunicaciones por proximidad (NFC) para la gestión de un estacionamiento público* [pdf] (tesis pregrado). Universidad Autónoma Metropolitana, Escuela de Ingeniería Electrónica, Iztapalapa, México. 2013. pp. 14-16. [Consulta: 2016-04-22]. Disponible en: http://tesiuami.izt.uam.mx/uam/aspuam/presentatesis.php?recno=16746&docs=UAMI16746.pdf

15. **SILBERSCHATZ, Abraham; KORTH, Henry; & SUDARSHAN, S.** *Fundamentos de Bases de Datos* [En línea]. 4ª ed. Madrid-España: McGRAW-HILL/INTERAMERICANA DE ESPAÑA, S. A. U, 2002. [Consulta: 22 febrero de 2016]. Disponible en: https://unefazuliasistemas.files.wordpress.com/2011/04/fundamentos-de-bases-de-datos-silberschatz-korth-sudarshan.pdf

16. **IGOE, Tom; et al.** *Beginning NFC Near Field Communication with Arduino, Android & Phonegap* [pdf]. 5 ª ed. Boston-USA: O'Reilly Media, 2014. [Consulta: 26 febrero de 2016]. Disponible en: ftp://facfiet.unicauca.edu.co/TalleresIoT/Libros/Beginning%20NFC%20-%20Near%20Field%20Communication%20with%20Arduino,%20Android,%20and%20PhoneGap%202014.pdf

17. **VEDAT, Coskun; KEREM, Ok; & BUSRA, Ozdenizci.** *NFC Application Development for Android* [en línea]. Estados Unidos: 2013. [Consulta: 27 febrero de 2016]. Disponible en: http://pdf.th7.cn/down/files/1312/professional_nfc_application_development_for_android.pdf?yundunkey=1ef6580cc35f41ec22395e20a294ff5a41430146400_415711093

18. **WHEAT, D.** *Arduino Internals* [en línea]. *Dallas, Texas, Estados Unidos.* Friendsof Apress. 2011. [Consulta: 25 febrero de 2016]. Disponible en: http://www.hmangas.com/Electronica/Datasheets/Arduino/LIBROS%20Y%20MANUALES/%5BArduino.Internals(2011)%5D.Dale.Wheat.pdf

ANEXOS:

Anexo A: Configuración Arduino (Procesamiento de Datos)

```
#include <SPI.h>
#include <Ethernet.h>
#include <Wire.h>
#include <PN532_SPI.h>
#include <NdefMessage.h>
#include <snep.h>

//------------------------Declaración Variables y Buffers-------------------------
PN532_SPI pn532spi(SPI, 53);
SNEP nfc(pn532spi);

uint8_t ndefBuf[200];
char inBuffer[1024];
String payloadAsString = "";
boolean startRead = false;
byte mac[] = {0xDE, 0xAD, 0xBE, 0xEF, 0xFE, 0xED};
char server[] = "192.168.1.2";
//IPAddress server(192,168,1,2);

IPAddress ip(192,168,1,20);
EthernetClient client;

void setup() {
Serial.begin(9600);
Serial.print("Configurando Shield Ethernet ");
Serial.println(".....");
delay(1000);
if (Ethernet.begin(mac) == 0) {
Serial.println("Fallo al configurar Ethernet usando DHCP");
Ethernet.begin(mac, ip);// Intenta configurar IP estatica en lugar de DHCP
}
```

```
Serial.print("IP:");
Serial.println(Ethernet.localIP());
delay(1000);
Serial.println("connecting...");
}

void loop() {
Serial.println("Ingresar ID:");
while(!Serial.available());
 if (Serial.available()>0){
 int identificador=(Serial.read()-48);
 httpRequest(identificador);
 delay(2000);
 Serial.println("Send a message to Peer");
 }
 delay(1000);
 Serial.flush();
 while(Serial.available()) Serial.read();
}
//



////----------------------------    NFC Recive packet  ----------------------------
String nfcRecive(){
int msgSize = nfc.read(ndefBuf, sizeof(ndefBuf));
if (msgSize > 0){
NdefMessage msg  = NdefMessage(ndefBuf, msgSize);
//msg.print();
NdefRecord record = msg.getRecord(0);
int payloadLength = record.getPayloadLength();
byte payload[payloadLength];
record.getPayload(payload);
// The TNF and Type are used to determine how your application processes the payload
// There's no generic processing for the payload, it's returned as a byte[]
int startChar = 0;
if (record.getTnf() == TNF_WELL_KNOWN && record.getType() == "T") { // text message
// skip the language code
```

```
startChar = payload[0] + 1;
} else if (record.getTnf() == TNF_WELL_KNOWN && record.getType() == "U") { // URI
// skip the url prefix (future versions should decode)
startChar = 1;
}
// Force the data into a String (might fail for some content)
// Real code should use smarter processing
payloadAsString = "";
for (int c = startChar; c < payloadLength; c++) {
payloadAsString += (char)payload[c];
}
}
return(payloadAsString);
}



//----------------------------     NFC Send packet  ----------------------------
char nfcSendPayload(char inBuff){
Serial.println("Send a message to Peer");
NdefMessage message = NdefMessage();
Serial.println(inBuffer);
message.addTextRecord(inBuffer);
//message.addMimeMediaRecord("text/plain", contenido);
int messageSize = message.getEncodedSize();
if (messageSize > sizeof(ndefBuf)) {
Serial.println("ndefBuf is too small");
while (1){}
}

message.encode(ndefBuf);
if (0 >= nfc.write(ndefBuf, messageSize)) {
Serial.println("Failed");
} else {
Serial.println("Success");
```

```
//        memset(ndefBuf,0,200);
}
delay(3000);
}


//----------------------------      HTTP Request  ----------------------------
void httpRequest(int identificador) {
if (client.connect(server, 80)) {
Serial.println("Conectado");
client.print("GET /subir_archivo/server_arduino.php?id=");
client.print(identificador);
client.println();
}
else {
Serial.println("Conexión fallida");
client.stop();
}
delay(500);
readPage();
}


//----------------------------     Reading file parts    ----------------------------
void readPage(){
int stringPos = 0;
boolean erroFlag=true;
memset(inBuffer, 0,1024);
while (client.available()){
char c = client.read();
if (c == '*' ) { //'*'begining character
startRead = true; //Ready to start the chunck
}else if(startRead){
if(c != '-'){ //'-'ending character
inBuffer[stringPos] = c;
stringPos ++;
}else
```

```
{
//send flag/clean client's buffer,inBuffer and count
startRead = false;
boolean flagArduino=true;
//  Serial.println(inBuffer);
nfcSendPayload(*inBuffer);
memset(inBuffer,0,1024);
stringPos = 0;
client.flush();
client.print(flagArduino);
client.println();
}
}
}
client.stop();
client.flush();
Serial.println("Desconectado");
}
```

ANEXO B: Configuración Android Studio (Interfaz de Usuario Final)

Actividad principal

```
package com.example.fie.info_fie;

import android.content.ActivityNotFoundException;
import android.content.Intent;
import android.net.Uri;
import android.nfc.NdefMessage;
import android.nfc.NdefRecord;
import android.nfc.NfcAdapter;
import android.os.Build;
import android.os.Bundle;
import android.support.design.widget.NavigationView;
import android.support.v4.app.ActivityCompat;
import android.support.v4.app.ActivityOptionsCompat;
import android.support.v4.app.Fragment;
import android.support.v4.util.Pair;
import android.support.v4.view.GravityCompat;
import android.support.v4.widget.DrawerLayout;
import android.support.v7.app.ActionBarDrawerToggle;
import android.support.v7.app.AppCompatActivity;
import android.support.v7.widget.Toolbar;
import android.view.MenuItem;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ImageView;
import android.widget.ListView;
import android.widget.TextView;
import android.widget.Toast;

import java.nio.charset.Charset;
import java.util.Locale;

public class MainActivity extends AppCompatActivity implements
NavigationView.OnNavigationItemSelectedListener
        ,OnListItemClickListener
        ,FragmentGallery.OnFragmentInteractionListener
        ,FragmentDocuments.OnFragmentInteractionListener
        ,FragmentReceive.OnFragmentInteractionListener
        ,AdapterView.OnItemClickListener{

    private NfcAdapter mNfcAdapter;
    private NdefMessage mNdefMessage;
    private String nfcvar="";
    private ListView lista;
    private ImageView imagen,banner;
    private TextView titulo, detalle;
    public boolean FragmentControl=false;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
        Toolbar toolbar = (Toolbar) findViewById(R.id.toolbar);
```

```
        setSupportActionBar(toolbar);

        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        ActionBarDrawerToggle toggle = new ActionBarDrawerToggle(
            this, drawer, toolbar, R.string.navigation_drawer_open,
R.string.navigation_drawer_close);
        drawer.addDrawerListener(toggle);
        toggle.syncState();

        NavigationView navigationView = (NavigationView)
findViewById(R.id.nav_view);
        navigationView.setNavigationItemSelectedListener(this);

        mNfcAdapter = NfcAdapter.getDefaultAdapter(this);
        Fragment fragmentInicio;
        fragmentInicio=new FragmentPantallaInicio();

getSupportFragmentManager().beginTransaction().replace(R.id.content_ma
in,fragmentInicio)
                .commit();


    }

    //----------------- Selecciona las opciones del menu --------------
--
    @Override
    public boolean onNavigationItemSelected(MenuItem item) {
        // Handle navigation view item clicks here.
        int id = item.getItemId();
        boolean FragmentTransaction=false;
        Fragment fragment=null;
        if (id == R.id.nav_information) {
            fragment=new FragmentGallery();
            FragmentTransaction=true;
            FragmentControl=false;

        } else if (id == R.id.nav_documents) {
            fragment=new FragmentDocuments();
            FragmentTransaction=true;
            FragmentControl=true;


        } else if (id == R.id.nav_receive) {
            Intent intent = new Intent(MainActivity.this,
ActivityReceive.class);
            startActivity(intent);

        } else if (id == R.id.nav_gallery) {
            Intent intent = new Intent(MainActivity.this,
ShowGalleryActivity.class);
            startActivity(intent);

        } else if (id == R.id.nav_maps) {
            fragment=new FragmentMaps();
            FragmentTransaction=true;
            FragmentControl=true;

        }else if (id == R.id.nav_share) {
```

```
            Intent myIntent=new Intent(Intent.ACTION_SEND);
            myIntent.setType("text/plain");
            String
shareBody="https://play.google.com/store/apps/details?id=se.anyro.nfc_
reader&hl=es";
            String ShareSub="Info_FIE";
            myIntent.putExtra(Intent.EXTRA_SUBJECT,ShareSub);
            myIntent.putExtra(Intent.EXTRA_TEXT,shareBody);
            startActivity(Intent.createChooser(myIntent,"Share
using"));
        }else if (id == R.id.nav_send) {
            try {
                Intent emailIntent = new Intent(Intent.ACTION_SEND);
                emailIntent.putExtra(Intent.EXTRA_EMAIL, new
String[]{"rjavier001@gmail.com"});
                emailIntent.putExtra(Intent.EXTRA_SUBJECT,
"Feedback");
                emailIntent.putExtra(Intent.EXTRA_TEXT, "Message!");
                emailIntent.setType("message/rfc822");
                startActivity(emailIntent);
            }catch (ActivityNotFoundException anfe){
                Toast toast=Toast.makeText(MainActivity.this, "No
email client found", Toast.LENGTH_SHORT);
                toast.show();
            }
        }

        if (FragmentTransaction){

getSupportFragmentManager().beginTransaction().replace(R.id.content_ma
in,fragment)
            .commit();
            item.setChecked(true);
            getSupportActionBar().setTitle(item.getTitle());
        }

        DrawerLayout drawer = (DrawerLayout)
findViewById(R.id.drawer_layout);
        drawer.closeDrawer(GravityCompat.START);
        return true;
    }


    @Override
    public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
        facultad item = (facultad) parent.getItemAtPosition(position);
        Intent intent = new Intent(this, ActivityGalleryDetail.class);
        intent.putExtra(ActivityGalleryDetail.EXTRA_PARAM_ID,
item.getId());
        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.LOLLIPOP) {
            ActivityOptionsCompat activityOptions =

ActivityOptionsCompat.makeSceneTransitionAnimation(
                            this,
                            new
Pair(view.findViewById(R.id.imagen_facultad),

ActivityGalleryDetail.VIEW_NAME_HEADER_IMAGE)
                    );
```

```
            ActivityCompat.startActivity(this, intent,
activityOptions.toBundle());
        } else
            startActivity(intent);

    }

    //---------------- Crea los mensajes NDEF ----------------
    public static NdefRecord createNewTextRecord(String text, Locale
locale, boolean encodeInUtf8) {
        byte[] langBytes =
locale.getLanguage().getBytes(Charset.forName("US-ASCII"));
        Charset utfEncoding = encodeInUtf8 ? Charset.forName("UTF-8")
: Charset.forName("UTF-16");
        byte[] textBytes = text.getBytes(utfEncoding);

        int utfBit = encodeInUtf8 ? 0 : (1 << 7);

        char status = (char)(utfBit + langBytes.length);
        byte[] data = new byte[1 + langBytes.length +
textBytes.length];
        data[0] = (byte)status;
        System.arraycopy(langBytes, 0, data, 1, langBytes.length);
        System.arraycopy(textBytes, 0, data, 1 + langBytes.length,
textBytes.length);

        return new NdefRecord(NdefRecord.TNF_WELL_KNOWN,
NdefRecord.RTD_TEXT, new byte[0], data);
    }
    @Override
    public void onListClick(int i) {
        imagen = (ImageView) findViewById(R.id.imageView_imagen);
        titulo = (TextView)  findViewById(R.id.textView_titulodoc);
        detalle = (TextView) findViewById(R.id.textView_detalle);

        switch (i){
            case 0:
                nfcvar = getString(R.string.idsend_doc1);
                imagen.setImageResource(R.drawable.ic_document);
                titulo.setText(getText(R.string.title_doc1));
                detalle.setText(getText(R.string.description_doc1));
                break;
            case 1:
                nfcvar = getString(R.string.idsend_doc2);
                imagen.setImageResource(R.drawable.ic_document);
                titulo.setText(getText(R.string.title_doc2));
                detalle.setText(getText(R.string.description_doc2));
                break;

            case 2:
                nfcvar = getString(R.string.idsend_doc3);
                imagen.setImageResource(R.drawable.ic_document);
                titulo.setText(getText(R.string.title_doc3));
                detalle.setText(getText(R.string.description_doc3));
                break;
            case 3:
                nfcvar = getString(R.string.idsend_doc4);
                imagen.setImageResource(R.drawable.ic_document);
                titulo.setText(getText(R.string.title_doc4));
                detalle.setText(getText(R.string.description_doc4));
```

```
            break;
        case 4:
            nfcvar = getString(R.string.idsend_doc5);
            imagen.setImageResource(R.drawable.ic_document);
            titulo.setText(getText(R.string.title_doc5));
            detalle.setText(getText(R.string.description_doc5));
            break;
        case 5:
            nfcvar = getString(R.string.idsend_doc6);
            imagen.setImageResource(R.drawable.ic_document);
            titulo.setText(getText(R.string.title_doc6));
            detalle.setText(getText(R.string.description_doc6));
            break;
        case 6:
            nfcvar = getString(R.string.idsend_doc7);
            imagen.setImageResource(R.drawable.ic_document);
            titulo.setText(getText(R.string.title_doc7));
            detalle.setText(getText(R.string.description_doc7));
            break;
        case 7:
            nfcvar = getString(R.string.idsend_doc8);
            imagen.setImageResource(R.drawable.ic_document);
            titulo.setText(getText(R.string.title_doc8));
            detalle.setText(getText(R.string.description_doc8));
            break;
        case 8:
            nfcvar = getString(R.string.idsend_doc9);
            imagen.setImageResource(R.drawable.ic_document);
            titulo.setText(getText(R.string.title_doc9));
            detalle.setText(getText(R.string.description_doc9));
            break;
        case 9:
            nfcvar = getString(R.string.idsend_doc10);
            imagen.setImageResource(R.drawable.ic_document);
            titulo.setText(getText(R.string.title_doc10));
            detalle.setText(getText(R.string.description_doc10));
            break;
        case 10:
            nfcvar = getString(R.string.idsend_doc11);
            imagen.setImageResource(R.drawable.ic_document);
            titulo.setText(getText(R.string.title_doc11));
            detalle.setText(getText(R.string.description_doc11));
            break;
        case 11:
            nfcvar = getString(R.string.idsend_doc12);
            imagen.setImageResource(R.drawable.ic_document);
            titulo.setText(getText(R.string.title_doc12));
            detalle.setText(getText(R.string.description_doc12));
            break;
        case 12:
            nfcvar = getString(R.string.idsend_doc13);
            imagen.setImageResource(R.drawable.ic_document);
            titulo.setText(getText(R.string.title_doc13));
            detalle.setText(getText(R.string.description_doc13));
            break;
        case 13:
            nfcvar = getString(R.string.idsend_doc14);
            imagen.setImageResource(R.drawable.ic_document);
            titulo.setText(getText(R.string.title_doc14));
            detalle.setText(getText(R.string.description_doc14));
```

```
                break;
        }

        mNdefMessage = new NdefMessage(
                new NdefRecord[]{createNewTextRecord(nfcvar,
Locale.ENGLISH, true)});

        if (mNfcAdapter != null && FragmentControl) {

            mNfcAdapter.setNdefPushMessage(mNdefMessage,
MainActivity.this);

        }
    }

    @Override
    public void onFragmentInteraction(Uri uri) {
    }
}
```

Fragment Documents

```
package com.example.fie.info_fie;

import android.app.Activity;
import android.content.Context;
import android.net.Uri;
import android.os.Bundle;
import android.support.v4.app.Fragment;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.ArrayAdapter;
import android.widget.ListView;
import android.widget.SearchView;
import android.widget.Toast;

public class FragmentDocuments extends Fragment implements
AdapterView.OnItemClickListener {
    private OnFragmentInteractionListener mListener;
    private ListView lista;
    private android.widget.SearchView busqueda;
    Activity activity;
    OnListItemClickListener listacallback;

    public FragmentDocuments() {
        // Required empty public constructor
    }

    @Override
    public void onAttach(Context context) {
        super.onAttach(context);
        activity = getActivity();
        try {
            listacallback = (OnListItemClickListener) getActivity();
        } catch (ClassCastException e) {
            throw new ClassCastException(activity.toString() + "ERROR
CALLBACK");
```

```
        }
    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
        View root =
inflater.inflate(R.layout.fragment_fragment_documents, container,
false);
        lista = (ListView) root.findViewById(R.id.listView);

busqueda=(android.widget.SearchView)root.findViewById(R.id.SearchView1
);
        String[] listadoc =
getResources().getStringArray(R.array.lista_doc);
        final ArrayAdapter<String> adapter = new
ArrayAdapter<String>(activity, android.R.layout.simple_list_item_1,
listadoc);
        lista.setAdapter(adapter);
        lista.setOnItemClickListener(this);
        busqueda.setOnQueryTextListener(new
SearchView.OnQueryTextListener() {
            @Override
            public boolean onQueryTextSubmit(String text) {
                return false;
            }

            @Override
            public boolean onQueryTextChange(String text) {
                adapter.getFilter().filter(text);
                return false;
            }
        });
        Toast.makeText(getActivity(), "ACERQUE el dispositivo y TOQUE
la pantalla para descargar el elemento seleccionado",
Toast.LENGTH_SHORT).show();
        return root;
    }


    @Override
    public void onItemClick(AdapterView<?> parent, View view, int
position, long id) {
        listacallback.onListClick(position);
    }

    public interface OnFragmentInteractionListener {
        // TODO: Update argument type and name
        void onFragmentInteraction(Uri uri);
    }
}
```

Activity Recibir

```
package com.example.fie.info_fie;

import android.app.PendingIntent;
import android.app.ProgressDialog;
import android.content.Context;
```

```
import android.content.DialogInterface;
import android.content.Intent;
import android.content.IntentFilter;
import android.content.res.Resources;
import android.nfc.NdefMessage;
import android.nfc.NdefRecord;
import android.nfc.NfcAdapter;
import android.os.AsyncTask;
import android.os.Bundle;
import android.os.Parcelable;
import android.support.v7.app.AppCompatActivity;
import android.util.Log;
import android.widget.TextView;
import android.widget.Toast;

import com.bumptech.glide.load.engine.Resource;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;

public class ActivityReceive extends AppCompatActivity {
    private TextView Content, contenido;
    private NfcAdapter nfcAdapter;
    private ProgressDialog pDialog;
    private MiTareaAsincronaDialog tarea;
    private int res;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_receive);
        nfcAdapter = NfcAdapter.getDefaultAdapter(this);
        contenido = (TextView) findViewById(R.id.txtdocs);
        Content = (TextView) findViewById(R.id.txtContent);
        if (nfcAdapter == null) {
            Toast.makeText(ActivityReceive.this,
                    "nfcAdapter==null, no NFC adapter exists",
                    Toast.LENGTH_LONG).show();
        } else {
            Content.setText("Waiting for NDEF Message");
        }
    }

    @Override
    protected void onResume() {
        super.onResume();

        enableForegroundDispatchSystem();
    }

    @Override
    protected void onPause() {
        super.onPause();

        disableForegroundDispatchSystem();
    }

    @Override
    protected void onNewIntent(Intent intent) {
```

```
        super.onNewIntent(intent);

        if (intent.hasExtra(NfcAdapter.EXTRA_NDEF_MESSAGES)) {
            Toast.makeText(this, "NfcIntent!",
Toast.LENGTH_SHORT).show();
            Parcelable[] parcelableMessages =
intent.getParcelableArrayExtra(NfcAdapter.EXTRA_NDEF_MESSAGES);
            if (parcelableMessages != null &&
parcelableMessages.length > 0) {
                readTextFromMessage((NdefMessage)
parcelableMessages[0]);
            } else {
                Toast.makeText(this, "No NDEF messages found!",
Toast.LENGTH_SHORT).show();
            }


        }
    }

    private void readTextFromMessage(NdefMessage ndefMessage) {

        NdefRecord[] ndefRecords = ndefMessage.getRecords();

        if (ndefRecords != null && ndefRecords.length > 0) {

            NdefRecord ndefRecord = ndefRecords[0];

            String tagContent = getTextFromNdefRecord(ndefRecord);

            Content.setText(tagContent);

        } else {
            Toast.makeText(this, "No NDEF records found!",
Toast.LENGTH_SHORT).show();
        }

    }

    public String getTextFromNdefRecord(NdefRecord ndefRecord) {
        String tagContent = null;
        try {
            byte[] payload = ndefRecord.getPayload();
            String textEncoding = ((payload[0] & 128) == 0) ? "UTF-8"
: "UTF-16";
            int languageSize = payload[0] & 0063;
            tagContent = new String(payload, languageSize + 1,
                    payload.length - languageSize - 1, textEncoding);
        } catch (UnsupportedEncodingException e) {
            Log.e("getTextFromNdefRecord", e.getMessage(), e);
        }
        return tagContent;
    }

    private void enableForegroundDispatchSystem() {

        Intent intent = new Intent(this,
ActivityReceive.class).addFlags(Intent.FLAG_RECEIVER_REPLACE_PENDING);

        PendingIntent pendingIntent = PendingIntent.getActivity(this,
```

```
 0, intent, 0);

        IntentFilter[] intentFilters = new IntentFilter[]{};

        nfcAdapter.enableForegroundDispatch(this, pendingIntent,
intentFilters, null);
    }

    private void disableForegroundDispatchSystem() {
        nfcAdapter.disableForegroundDispatch(this);
    }

    private void tareaLarga() {
        try {
            Thread.sleep(650);
        } catch (InterruptedException e) {
        }
    }

    private void dialog() {
        pDialog = new ProgressDialog(ActivityReceive.this);
        pDialog.setProgressStyle(ProgressDialog.STYLE_HORIZONTAL);
        pDialog.setMessage("Receiving...");
        pDialog.setCancelable(true);
        pDialog.setMax(100);
        tarea = new MiTareaAsincronaDialog();
        tarea.execute();
    }

    private class MiTareaAsincronaDialog extends AsyncTask<Void,
Integer, Boolean> {

        @Override
        protected Boolean doInBackground(Void... params) {

            for (int i = 1; i <= 10; i++) {
                tareaLarga();

                publishProgress(i * 10);

                if (isCancelled())
                    break;
            }

            return true;
        }

        @Override
        protected void onProgressUpdate(Integer... values) {
            int progreso = values[0].intValue();

            pDialog.setProgress(progreso);
        }

        @Override
        protected void onPreExecute() {
            pDialog.setOnCancelListener(new
DialogInterface.OnCancelListener() {
                @Override
                public void onCancel(DialogInterface dialog) {
                    MiTareaAsincronaDialog.this.cancel(true);
```

```
                }
            });
            pDialog.setProgress(0);
            pDialog.show();
        }

        @Override
        protected void onPostExecute(Boolean result) {
            if (result) {
                pDialog.dismiss();
                Toast.makeText(ActivityReceive.this, "Descarga
finalizada!", Toast.LENGTH_SHORT).show();
                leerArchivo(res);
            }
        }

        @Override
        protected void onCancelled() {
            Toast.makeText(ActivityReceive.this, "Tarea cancelada!",
Toast.LENGTH_SHORT).show();
        }
    }


    private void leerArchivo(int document) {
        try {
                    InputStreamReader isr = new
InputStreamReader(this.getResources().openRawResource(document));
                    BufferedReader br = new BufferedReader(isr);
                    String linea;
                    StringBuilder texto = new StringBuilder();
                    while ((linea = br.readLine()) != null) {
                        texto.append(linea);
                        texto.append("\n");
                    }
                    br.close();
                    isr.close();
                    contenido.setText(texto.toString());
        } catch (IOException e) {
            e.printStackTrace();
        }
    }
}
```

ANEXO C: Configuración PHP (Base de Datos)

/Descargar Archivo

```
<?php
require("dbconnect.inc.php");
if ($_SERVER['REQUEST_METHOD'] == 'POST')
{
$id= $_POST['id'];
$sql = "SELECT tipo, contenido FROM archivos WHERE id='$id'";
// $data = mysql_query($sql) or die(mysql_error());
// $documento = mysql_fetch_object($data);
// print($documento);
$res=mysql_query($sql);
$tipo=mysql_result($res, 0, "tipo");
$contenido=mysql_result($res, 0, "contenido");
header("Content-type: $tipo");
print($contenido);
exit;
}
?>
<form method='POST'>
<input type='text' name='id'>
<input type='submit'>
</form>
```

/Guardar archivo

```
<?php
require("dbconnect.inc.php");
formatos = array('.txt');
$archivoTemp=$_FILES["archivo"]["tmp_name"];
$tamanio=$_FILES["archivo"]["size"];
```

```
$tipo=$_FILES["archivo"]["type"];
$nombre=$_FILES["archivo"]["name"];
$titulo=$_POST["titulo"];
$descripcion=$_POST["descripcion"];
$ext=substr($nombre, strripos($nombre,'.'));
if(in_array($ext, $formatos)){
if ( $archivoTemp != "none" ){
$fp = fopen($archivoTemp, "rb");
$contenido = fread($fp, $tamanio);
$contenido = addslashes($contenido);
fclose($fp);
$qry = "INSERT INTO archivo VALUES
(0,'$nombre','$titulo','$descripcion','$contenido','$tipo')";
mysql_query($qry);
if(mysql_affected_rows($con) > 0)
print "Se ha guardado el archivo en la base de datos.";
else
print "NO se ha podido guardar el archivo en la base de datos."; }
else
print "No se ha podido subir el archivo al servidor";
}else{
echo "ARCHIVO NO PERMITIDO";}
?>
```

/Petición al servidor

```
<?php
//   setting BD
$conexion = mysql_connect("localhost","jada","12345");
mysql_select_db("bd_repositorio",$conexion);
$resultado = mysql_query("SELECT * FROM archivo  WHERE id_doc='" . $_GET['id']. "'",
$conexion);
$contenido=mysql_result($resultado, 0, "contenido");
```

```
//_____
_____
//   setting the unlimited memory for reading huge pdf
ini_set("max_execution_time",-1);
// reading a pdf file
// $file = file_get_contents($contenido);
$string_array = str_split(base64_encode($contenido),128);
// $string_array = str_split($file,16);
$numFrames=count($string_array);
// echo "*".$numFrames."-";
$flagArduino=true;
$pos_ini=0;
$x=0;
while ($numFrames>0) {
if ($flagArduino=true) {
$framestoRead=min($numFrames,1);
$pos_ini=0+$x;
$pos_fin=$framestoRead+$x;
echo "*";
for ($i=$pos_ini; $i<=$pos_fin-1; $i++) {
// echo"< ".$string_array[$i]." >";
echo $string_array[$i];
$x++; }
echo "-";
$flagArduino=false;
$numFrames-=$framestoRead; }
if($numFrames!=0){
 // echo "<br>";
 // $flagArduino=$_GET['flagArduino'];
 if (!isset($_GET['flagArduino'])) {
 $flagArduino=$_GET['flagArduino'];
 }         }         }
```

```
 ?>
```

/ Conexión

```
<?php
$conexion = mysql_connect("localhost","jada","12345");
mysql_select_db("bd_repositorio",$conexion);
$resultado = mysql_query("SELECT * FROM archivo  WHERE id_doc='" . $_GET['id']. "'", $conexion);
$valor=mysql_result($resultado, 0, "titulo");
echo "valor=" . $valor . ";";
?>
```

/Listar archivo

```
<!DOCTYPE html>
<html>
<head>
<meta charset="UTF-8">
<title></title>
</head>
<body>
<table>
<tr>
<td>NOMBRE</td>
<td>TITULO</td>
<td>DESCRICION</td>
<td>CONTENIDO</td>
<td>TIPO</td>
</tr>
<?php
$con = mysqli_connect("localhost","jada","12345");
mysqli_select_db($con,"bd_repositorio");
$query = mysqli_query($con, "SELECT * FROM archivo");
```

```
$tildes = $con->query("SET NAMES 'utf8'");
while($datos=mysqli_fetch_array($query)){?>
<tr>
<td><?php echo $datos['nombre']; ?></td>
<td><?php echo $datos['titulo']; ?></td>
<td><?php echo $datos['descripcion']; ?></td>
<!-- <td><?php echo $datos['contenido']; ?></td>  -->
<td><?php echo $datos['tipo']; ?></td>
<!-- <td><a href="archivo.php?id=<?php echo $datos['id_doc']?>"><?php echo $datos['nombre']; ?></a></td> -->
</tr>
<?php  } ?>
</table>
</body>
</html>
```

ANEXO D: (Datasheet Arduino Uno)

Arduino Uno

Arduino Uno R3 Front

Arduino Uno R3 Back


Arduino Uno R2 Front


Arduino Uno SMD


Arduino Uno Front


Arduino Uno Back

Overview

The Arduino Uno is a microcontroller board based on the ATmega328 (datasheet). It has 14 digital input/output pins (of which 6 can be used as PWM outputs), 6 analog inputs, a 16 MHz ceramic resonator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started.
The Uno differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega16U2 (Atmega8U2 up to version R2) programmed as a USB-to-serial converter.
Revision 2 of the Uno board has a resistor pulling the 8U2 HWB line to ground, making it easier to put into DFU mode.
Revision 3 of the board has the following new features:

- 1.0 pinout: added SDA and SCL pins that are near to the AREF pin and two other new pins placed near to the RESET pin, the IOREF that allow the shields to adapt to the voltage provided from the board. In future, shields will be compatible both with the board that use the AVR, which operate with 5V and with the Arduino Due that operate with 3.3V. The second one is a not connected pin, that is reserved for future purposes.
- Stronger RESET circuit.
- Atmega 16U2 replace the 8U2.

"Uno" means one in Italian and is named to mark the upcoming release of Arduino 1.0. The Uno and version 1.0 will be the reference versions of Arduino, moving forward. The Uno is the latest in a series of USB Arduino boards, and the reference model for the Arduino platform; for a comparison with previous versions, see the index of Arduino boards.

Summary

Microcontroller	ATmega328
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	14 (of which 6 provide PWM output)
Analog Input Pins	6
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (ATmega328) of which 0.5 KB used by bootloader
SRAM	2 KB (ATmega328)
EEPROM	1 KB (ATmega328)
Clock Speed	16 MHz

Schematic & Reference Design

EAGLE files: arduino-uno-Rev3-reference-design.zip (NOTE: works with Eagle 6.0 and newer)
Schematic: arduino-uno-Rev3-schematic.pdf
Note: The Arduino reference design can use an Atmega8, 168, or 328, Current models use an ATmega328, but an Atmega8 is shown in the schematic for reference. The pin configuration is identical on all three processors.

Power

The Arduino Uno can be powered via the USB connection or with an external power supply. The power source is selected automatically.
External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.
The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.
The power pins are as follows:

- **VIN.** The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- **5V.** This pin outputs a regulated 5V from the regulator on the board. The board can be supplied with power either from the DC power jack (7 - 12V), the USB connector (5V), or the VIN pin of the board (7-12V). Supplying voltage via the 5V or 3.3V pins bypasses the regulator, and can damage your board. We don't advise it.
- **3V3.** A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- **GND.** Ground pins.

Memory

The ATmega328 has 32 KB (with 0.5 KB used for the bootloader). It also has 2 KB of SRAM and 1 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 14 digital pins on the Uno can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- **Serial: 0 (RX) and 1 (TX).** Used to receive (RX) and transmit (TX) TTL serial data. These pins are connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip.
- **External Interrupts: 2 and 3.** These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- **PWM: 3, 5, 6, 9, 10, and 11.** Provide 8-bit PWM output with the analogWrite() function.
- **SPI: 10 (SS), 11 (MOSI), 12 (MISO), 13 (SCK).** These pins support SPI communication using the SPI library.
- **LED: 13.** There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.

The Uno has 6 analog inputs, labeled A0 through A5, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and the analogReference() function. Additionally, some pins have specialized functionality:

- **TWI: A4 or SDA pin and A5 or SCL pin.** Support TWI communication using the Wire library.

There are a couple of other pins on the board:

- **AREF.** Reference voltage for the analog inputs. Used with analogReference().
- **Reset.** Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

See also the mapping between Arduino pins and ATmega328 ports. The mapping for the Atmega8, 168, and 328 is identical.

Communication

The Arduino Uno has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega328 provides UART TTL (5V) serial communication, which is available on digital pins 0 (RX) and 1 (TX). An ATmega16U2 on the board channels this serial communication over USB and appears as a virtual com port to software on the computer. The '16U2 firmware uses the standard USB COM drivers, and no external driver is needed. However, on Windows, a .inf file is required. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the Arduino board. The RX and TX LEDs on the board will flash when data is being transmitted via the USB-to-serial chip and USB connection to the computer (but not for serial communication on pins 0 and 1).
A SoftwareSerial library allows for serial communication on any of the Uno's digital pins.
The ATmega328 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation for details. For SPI communication, use the SPI library.

Programming

The Arduino Uno can be programmed with the Arduino software (download). Select "Arduino Uno from the **Tools > Board** menu (according to the microcontroller on your board). For details, see the reference and tutorials.
The ATmega328 on the Arduino Uno comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).
You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.
The ATmega16U2 (or 8U2 in the rev1 and rev2 boards) firmware source code is available . The ATmega16U2/8U2 is loaded with a DFU bootloader, which can be activated by:

- On Rev1 boards: connecting the solder jumper on the back of the board (near the map of Italy) and then resetting the 8U2.
- On Rev2 or later boards: there is a resistor that pulling the 8U2/16U2 HWB line to ground, making it easier to put into DFU mode.

You can then use Atmel's FLIP software (Windows) or the DFU programmer (Mac OS X and Linux) to load a new firmware. Or you can use the ISP header with an external programmer (overwriting the DFU bootloader). See this user-contributed tutorial for more information.

Automatic (Software) Reset

Rather than requiring a physical press of the reset button before an upload, the Arduino Uno is designed in a way that allows it to be reset by software running on a connected computer. One of the hardware flow control lines (DTR) of the ATmega8U2/16U2 is connected to the reset line of the ATmega328 via a 100 nanofarad capacitor. When this line is asserted (taken low), the reset line drops long enough to reset the chip. The Arduino software uses this capability to allow you to upload code by simply pressing the upload button in the Arduino environment. This means that the bootloader can have a shorter timeout, as the lowering of DTR can be well-coordinated with the start of the upload.
This setup has other implications. When the Uno is connected to either a computer running Mac OS X or Linux, it resets each time a connection is made to it from software (via USB). For the following half-second or so, the bootloader is running on the Uno. While it is programmed to ignore malformed data (i.e. anything besides an upload of new code), it will intercept the first few bytes of data sent to the board after a connection is opened. If a sketch running on the board receives one-time configuration or other data when it first starts, make sure that the software with which it communicates waits a second after opening the connection and before sending this data.
The Uno contains a trace that can be cut to disable the auto-reset. The pads on either side of the trace can be soldered together to re-enable it. It's labeled "RESET-EN". You may also be able to disable the auto-reset by connecting a 110 ohm resistor from 5V to the reset line; see this forum thread for details.

USB Overcurrent Protection

The Arduino Uno has a resettable polyfuse that protects your computer's USB ports from shorts and overcurrent. Although most computers provide their own internal protection, the fuse provides an extra layer of protection. If more than 500 mA is applied to the USB port, the fuse will automatically break the connection until the short or overload is removed.

Physical Characteristics

The maximum length and width of the Uno PCB are 2.7 and 2.1 inches respectively, with the USB connector and power jack extending beyond the former dimension. Four screw holes allow the board to be attached to a surface or case. Note that the distance between digital pins 7 and 8 is 160 mil (0.16"), not an even multiple of the 100 mil spacing of the other pins.

ANEXO E: (Datasheet Arduino Mega 2560)

Arduino MEGA 2560

Product Overview

The Arduino Mega 2560 is a microcontroller board based on the ATmega2560 (datasheet). It has 54 digital input/output pins (of which 14 can be used as PWM outputs), 16 analog inputs, 4 UARTs (hardware serial ports), a 16 MHz crystal oscillator, a USB connection, a power jack, an ICSP header, and a reset button. It contains everything needed to support the microcontroller; simply connect it to a computer with a USB cable or power it with a AC-to-DC adapter or battery to get started. The Mega is compatible with most shields designed for the Arduino Duemilanove or Diecimila.

Index

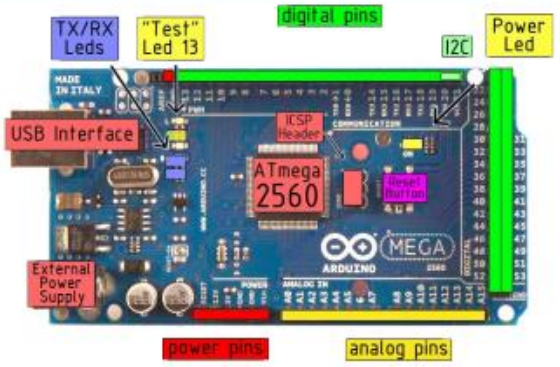
Technical Specification

EAGLE files: arduino-mega2560-reference-design.zip Schematic: arduino-mega2560-schematic.pdf

Summary

Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limits)	6-20V
Digital I/O Pins	54 (of which 14 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz

the board



Power

The Arduino Mega2560 can be powered via the USB connection or with an external power supply. The power source is selected automatically. External (non-USB) power can come either from an AC-to-DC adapter (wall-wart) or battery. The adapter can be connected by plugging a 2.1mm center-positive plug into the board's power jack. Leads from a battery can be inserted in the Gnd and Vin pin headers of the POWER connector.

The board can operate on an external supply of 6 to 20 volts. If supplied with less than 7V, however, the 5V pin may supply less than five volts and the board may be unstable. If using more than 12V, the voltage regulator may overheat and damage the board. The recommended range is 7 to 12 volts.

The Mega2560 differs from all preceding boards in that it does not use the FTDI USB-to-serial driver chip. Instead, it features the Atmega8U2 programmed as a USB-to-serial converter.

The power pins are as follows:

- VIN. The input voltage to the Arduino board when it's using an external power source (as opposed to 5 volts from the USB connection or other regulated power source). You can supply voltage through this pin, or, if supplying voltage via the power jack, access it through this pin.
- 5V. The regulated power supply used to power the microcontroller and other components on the board. This can come either from VIN via an on-board regulator, or be supplied by USB or another regulated 5V supply.
- 3V3. A 3.3 volt supply generated by the on-board regulator. Maximum current draw is 50 mA.
- GND. Ground pins.

Memory

The ATmega2560 has 256 KB of flash memory for storing code (of which 8 KB is used for the bootloader), 8 KB of SRAM and 4 KB of EEPROM (which can be read and written with the EEPROM library).

Input and Output

Each of the 54 digital pins on the Mega can be used as an input or output, using pinMode(), digitalWrite(), and digitalRead() functions. They operate at 5 volts. Each pin can provide or receive a maximum of 40 mA and has an internal pull-up resistor (disconnected by default) of 20-50 kOhms. In addition, some pins have specialized functions:

- Serial: 0 (RX) and 1 (TX); Serial 1: 19 (RX) and 18 (TX); Serial 2: 17 (RX) and 16 (TX); Serial 3: 15 (RX) and 14 (TX). Used to receive (RX) and transmit (TX) TTL serial data. Pins 0 and 1 are also connected to the corresponding pins of the ATmega8U2 USB-to-TTL Serial chip .
- External Interrupts: 2 (interrupt 0), 3 (interrupt 1), 18 (interrupt 5), 19 (interrupt 4), 20 (interrupt 3), and 21 (interrupt 2). These pins can be configured to trigger an interrupt on a low value, a rising or falling edge, or a change in value. See the attachInterrupt() function for details.
- PWM: 0 to 13. Provide 8-bit PWM output with the analogWrite() function.
- SPI: 50 (MISO), 51 (MOSI), 52 (SCK), 53 (SS). These pins support SPI communication, which, although provided by the underlying hardware, is not currently included in the Arduino language. The SPI pins are also broken out on the ICSP header, which is physically compatible with the Duemilanove and Diecimila.
- LED: 13. There is a built-in LED connected to digital pin 13. When the pin is HIGH value, the LED is on, when the pin is LOW, it's off.
- I²C: 20 (SDA) and 21 (SCL). Support I²C (TWI) communication using the Wire library (documentation on the Wiring website). Note that these pins are not in the same location as the I²C pins on the Duemilanove.

The Mega2560 has 16 analog inputs, each of which provide 10 bits of resolution (i.e. 1024 different values). By default they measure from ground to 5 volts, though is it possible to change the upper end of their range using the AREF pin and analogReference() function.

There are a couple of other pins on the board:

- AREF. Reference voltage for the analog inputs. Used with analogReference().
- Reset. Bring this line LOW to reset the microcontroller. Typically used to add a reset button to shields which block the one on the board.

Communication

The Arduino Mega2560 has a number of facilities for communicating with a computer, another Arduino, or other microcontrollers. The ATmega2560 provides four hardware UARTs for TTL (5V) serial communication. An ATmega8U2 on the board channels one of these over USB and provides a virtual com port to software on the computer (Windows machines will need a .inf file, but OSX and Linux machines will recognize the board as a COM port automatically. The Arduino software includes a serial monitor which allows simple textual data to be sent to and from the board. The RX and TX LEDs on the board will flash when data is being transmitted via the ATmega8U2 chip and USB connection to the computer (but not for serial communication on pins 0 and 1).

A SoftwareSerial library allows for serial communication on any of the Mega's digital pins.

The ATmega2560 also supports I2C (TWI) and SPI communication. The Arduino software includes a Wire library to simplify use of the I2C bus; see the documentation on the Wiring website for details. To use the SPI communication, please see the ATmega2560 datasheet.

Programming

The Arduino Mega2560 can be programmed with the Arduino software (download). For details, see the reference and tutorials.

The Atmega2560 on the Arduino Mega comes preburned with a bootloader that allows you to upload new code to it without the use of an external hardware programmer. It communicates using the original STK500 protocol (reference, C header files).

You can also bypass the bootloader and program the microcontroller through the ICSP (In-Circuit Serial Programming) header; see these instructions for details.

ANEXO F: (Datasheet Arduino Yún)

Arduino Yun

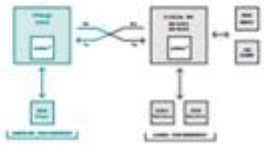
Overview

Arduino Yún is a microcontroller board based on the ATmega32u4 (datasheet) and the Atheros AR9331. The Atheros processor supports a Linux distribution based on OpenWrt named Linino OS. The board has built-in Ethernet and WiFi support, a USB-A port, micro-SD card slot, 20 digital input/output pins (of which 7 can be used as PWM outputs and 12 as analog inputs), a 16 MHz crystal oscillator, a micro USB connection, an ICSP header, and a 3 reset buttons.

The Yún distinguishes itself from other Arduino boards in that it can communicate with the Linux distribution onboard, offering a powerful networked computer with the ease of Arduino. In addition to Linux commands like cURL, you can write your own shell and python scripts for robust interactions.

The Yún is similar to the Leonardo in that the ATmega32u4 has built-in USB communication, eliminating the need for a secondary processor. This allows the Yún to appear to a connected computer as a mouse and keyboard, in addition to a virtual (CDC) serial / COM port.

The Bridge library facilitates communication between the two processors, giving Arduino sketches the ability to run shell scripts, communicate with network interfaces, and receive information from the AR9331 processor. The USB host, network interfaces and SD card are not connected to the 32U4, but the AR9331, and the Bridge library also enables the Arduino to interface with those peripherals.



Arduino Yun

Description

AVR Microcontroller

Microcontroller	ATmega32u4
Operating Voltage	5V
Input Voltage	5V
Digital I/O Pins	20
PWM Channels	7
Analog Input Channels	12
DC Current per I/O Pin	40 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	32 KB (of which 4 KB used by bootloader)
SRAM	2.5 KB
EEPROM	1 KB
Clock Speed	16 MHz

Linux microprocessor

Processor	Atheros AR9331
Architecture	MIPS @400MHz
Operating Voltage	3.3V
Ethernet	IEEE 802.3 10/100Mbit/s
WiFi	IEEE 802.11b/g/n
USB Type-A	2.0 Host
Card Reader	Micro-SD only
RAM	64 MB DDR2
Flash Memory	16 MB
PoE compatible 802.3af card support (see the note below)	
Length	73 mm
Width	53 mm
Weight	32 g

with PoE

without PoE